
SideFX®
Houdini™
FOUNDATIONS
FOR FILM, TV & GAMEDEV
VERSION 19.5 EDITION

Houdini™

FOUNDATIONS

FOR FILM, TV & GAMEDEV

VERSION 19.5 EDITION

ROBERT MAGEE

SideFX®

HOUDINI FOUNDATIONS

Author: Robert Magee
Research: Michael Buckley, Marcia Utama
Cover Art: Attila Torok

Special thanks to Kim Davidson, Richard Hamel, Chris Hebert, Cristin Barghiel
and everyone at SideFX who brings Houdini to life.

ISBN: 978-1-7753338-2-1

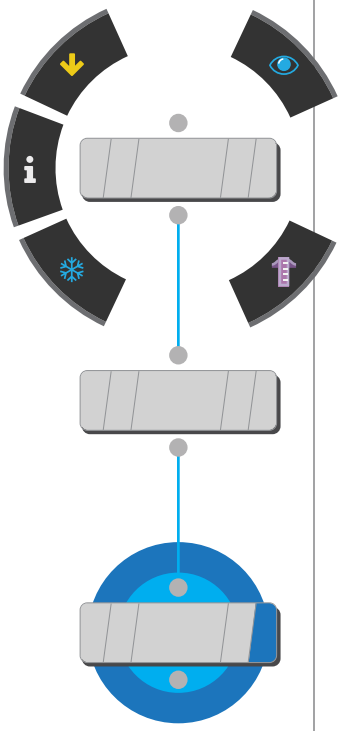
First Published in 2022
by SideFX Software
123 Front Street West, Suite 1401, Toronto, Ontario M5J 2M2

Written for the features found in Houdini 19.5
Document Version 2.0 | December 2022

HOUDINI FOUNDATIONS

CONTENTS:

HOUDINI FOUNDATIONS
OVERVIEW



To create 3D animation and VFX for Film, TV, Video Games and VR, you need a combination of technical and creative skills. Houdini is the perfect tool for bringing these worlds together as you explore, create and refine your projects from concept to final sign off.

While Houdini has a wide variety of tools designed for generating CG content, its **node-based procedural workflow** is what sets it apart. This approach makes it easier for you to **create directable shots**, explore **multiple iterations** and **hit deadlines**. As you learn Houdini, understanding how to work with these nodes and networks will be important to your success.

WHAT YOU WILL LEARN

This overview chapter contains general information about Houdini that will help you become familiar with important concepts and ideas. While you might not understand what it all means, this chapter will be a valuable reference point as you work through the foundation tutorials and build up your knowledge.

- **If you already work with 3D software** then learning Houdini will be a transfer of existing skills. You will learn how to interactively build-up shots using the scene view and shelf tools, then how to work with the nodes and networks to take advantage of Houdini's procedural nature.

- **If you are new to 3D and Computer Graphics** then Houdini is a great package to start with. The Foundations material assumes some general knowledge, therefore you may want to read up on CG concepts that you are unfamiliar with. In the end, Houdini will help you achieve a deeper understanding of what goes on under the hood of not only Houdini but other 3D apps as well.

Once you have finished the foundation lessons, visit **SideFX.com** for more tutorials. From the main menu, go to **Learn > Learning Paths** for a comprehensive list of lessons created by SideFX and members of the Houdini community. There are lots of lessons for you to explore as you build up your Houdini skillset.

DOWNLOAD
HOUDINI FOR FREE

SideFX has a free learning edition for you to use as you work through the lessons. The **Houdini Apprentice** edition gives you **FREE** access to all of Houdini's features with a few restrictions such as limited render size and user interface and render watermarks.

You can download Houdini Apprentice from the SideFX website where you can also get the latest versions which are updated regularly:

SideFX.com/download

INDIE
ANIMATORS AND GAMERS

If you want to go beyond the free learning edition, **Houdini Indie** removes the watermarks found in Apprentice and offers higher render resolutions up to 4K x 4K and limited commercial use **[less than $100K USD]** of Houdini.

The Indie program makes Houdini a great tool for developing personal projects and indie games. To learn more you can go to:

SideFX.com/indie

Document Version 4.0 | July 2022 | Houdini 19.5+

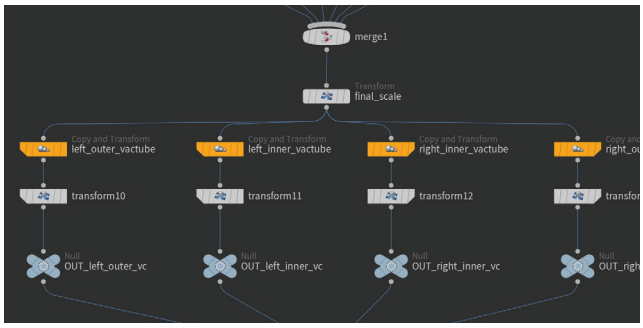
Learning Houdini

Houdini is a computer graphics, or CG, application which you can use to model, animate, render and simulate. In the process of learning Houdini, you will explore new ways of managing the creative process that involves the interactive manipulation of nodes, networks and assets.

Everything is procedural in Houdini, which means that modeling, character rigging, lighting, rendering and visual effects all benefit from a node-based workflow where artists can build up networks of nodes to manage all the steps needed to complete a creative task. Networks can then "talk" to other networks to create even more sophisticated systems.

GOING PROCEDURAL

In Houdini, every action you take is stored as a node. These nodes are then "wired" into networks which define a "recipe" that can be tweaked to define a repeatable outcome where each iteration can generate unique results. The ability for nodes to pass important information down the chain, in the form of attributes, helps give Houdini its procedural nature.



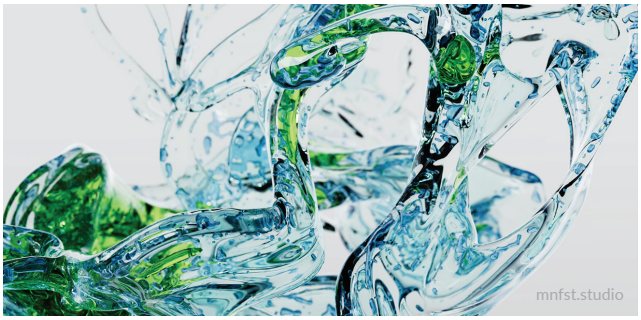
KNOWN FOR VFX

Visual effects artists have traditionally gravitated to Houdini because this procedural workflow is ideal for working with particles and dynamics. Often visual effects are designed to react to actions that are taking place in a shot and a procedural solutions "automate" these reactions. Therefore, Houdini provides studios with higher levels of productivity and more control over the creative process.

Houdini is also capable of working with large data sets which is critical as visual effects become more sophisticated with many layers such as rigid body destruction, fluids, and particles all interacting to achieve the final result.

PROCEDURAL CONSTRUCTS

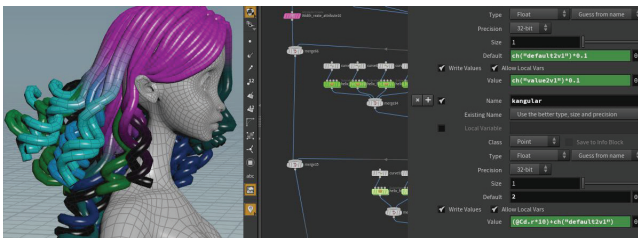
For motion graphics projects, a procedural approach offers lots of visually-stunning eye candy. These special effects are often the result of animating parameters on nodes and adding noise in interesting ways that you wouldn't expect in real life.



THE WIDER CG PIPELINE

Beyond VFX and motion graphics, Houdini has bread-and-butter tools for all parts of the pipeline from modeling to rendering to character work and gamedev. Its procedural workflow supports you as you create all of your CG content. Along the way, you will benefit from the ability to explore multiple iterations and make changes deep into production.

While the nodes are what makes Houdini unique and give it its power, there are lots of viewport and shelf tools that are used to work interactively while Houdini builds the networks.

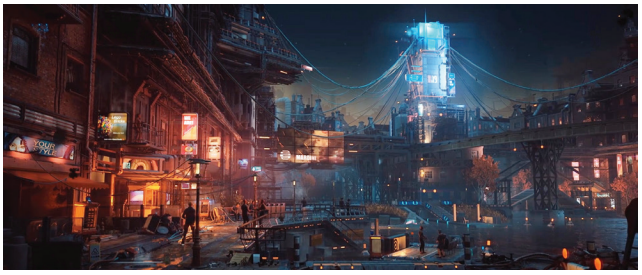


DIRECTABLE RESULTS

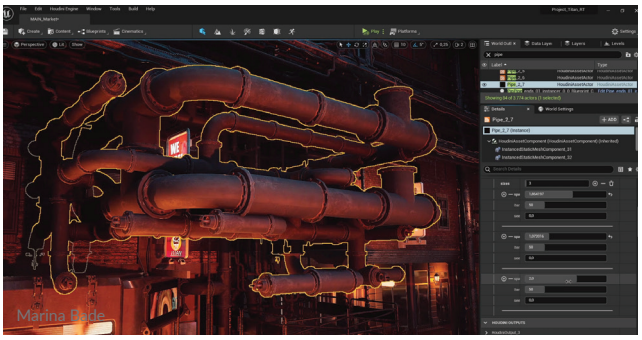
The reason you are able to make edits deep into production is because changes made to parameters on Houdini nodes will cascade right through the network to update the results. This directability is retained throughout the creative process and can be used to make last minute decisions that would be too costly in a traditional CG pipeline.

TOOL BUILDING

Another benefit of the node-based approach is that it is easy to encapsulate node networks to create custom nodes that are shared with colleagues without writing any code. Houdini's re-usable networks can be wrapped up quickly and easily into special nodes called **Houdini Digital Assets**.

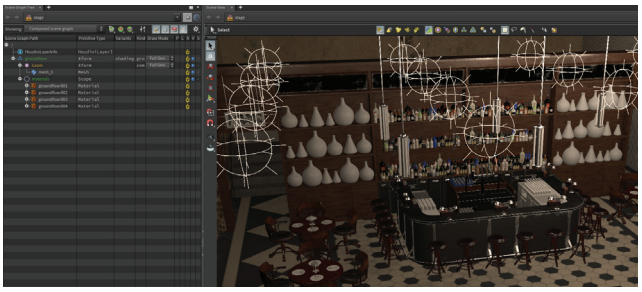


These assets open in Houdini, or in other applications such as **Autodesk Maya, 3DS Max**, **Unreal** and **Unity** using **Houdini Engine** plug-ins, with the asset's procedural nature left intact.



FULL ACCESS TO ALL YOUR DATA

As objects move through a typical animation or visual effects pipeline, they accumulate information which is often stored as point or primitive attributes such as velocity, capture weights or UV texture coordinates. While other 3D applications hide this information and attempt to control it for you behind the scenes, Houdini gives you tools for working with and managing this data. This results in a much more powerful and flexible approach that makes a huge difference down in production.



A NEW WAY OF THINKING

As you become more proficient with Houdini, you will find new ways to approach a shot or a game level that will make you and your team more productive. Houdini gives you the flexibility to build tools that will support you throughout a project's life cycle and instead of simply reacting to issues and problems, you will be able to anticipate the pain points and use a procedural solution to work much more efficiently.

Now that you have chosen to learn Houdini, you will find yourself exploring a versatile application that will redefine how you approach future projects. The key is to embrace this new way of working and be ready to explore CG at a level deeper than you ever imagined.

DO I NEED TO WRITE CODE TO USE HOUDINI?

DEFINITELY NOT! In fact, because of Houdini's node-based workflow, you will often be able to create results interactively that would require writing code in other 3D applications. Houdini is very much an artist's tool and while it has a technical side that uses scripts and expressions, the out-of-box tools will let you accomplish amazing things. And the nodes let you easily go back and make changes which mimics how the creative process works.

If you do want to work with code then Houdini supports a number of languages inside the Houdini interface. There are Wrangle nodes for working with VEX and Python and PyQT is supported as well. You can also use Houdini's expression language hscript or you can mix all of them together to meet your specific needs.

The Houdini Workspace

Houdini offers a user interface experience that will be familiar for artists coming from other CG applications with the biggest difference being the panes used to manage nodes and networks. The workspace is highly configurable and can be set up to support different ways of working.

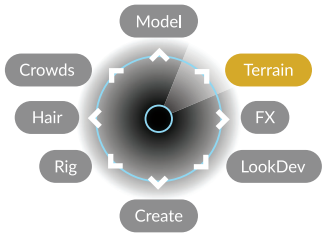
Houdini gives artists many different ways to view the bits and pieces that make up a 3D scene. From the **Scene View** where you look through a camera at your geometry to the **Network view** where you manage the procedural nodes and networks, you will find many different ways to make creative decisions while making sure each shot works at a technical level.

RADIAL MENUS

One way to access tools in the Scene View is the radial menus which you can access using the **X**, **C** and **V** hotkeys. Each of these brings up a radial menu with lots of options for you to choose from. The main focus of each menu is as follows:

- **Snapping** X
- **Main** C
- **Views** V

Once you learn how a radial menu works, you can access the tool with a quick sweep gesture without dwelling on the widget.



You can change the **custom** menu at the top of the menu bar which says **Main** by default. On OS X this is the **Radial** menu.

SHELF TOOLS

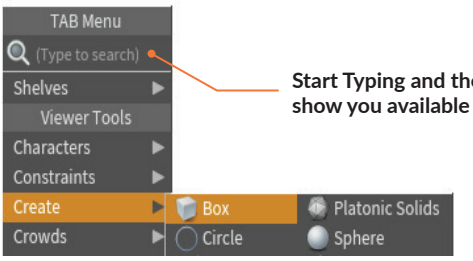
At the top of the workspace, you will find multiple shelves filled with tools for creating and manipulating objects, geometry, cameras, lights and effects.

These tools work in the scene view and often involve some sort of scene view interaction. Once you have used one of these tools, one or more nodes will be created which you can then refine in the **Parameter** and **Network** panes.

The shelves provide a very important resource to new Houdini artists because the shelves reduce clicks and often put down networks of nodes that you can learn from.

TAB MENU

Another way to access tools in either the **Scene** view or the **Network** view is to press the **tab** key. This brings up a menu of available tools and nodes you can use in your work.



Tool Shelf - The shelf tools let you work with objects and geometry in the scene view.

TOOL BAR:

Selection Modes - Focus on scene, geometry, or dynamic objects.

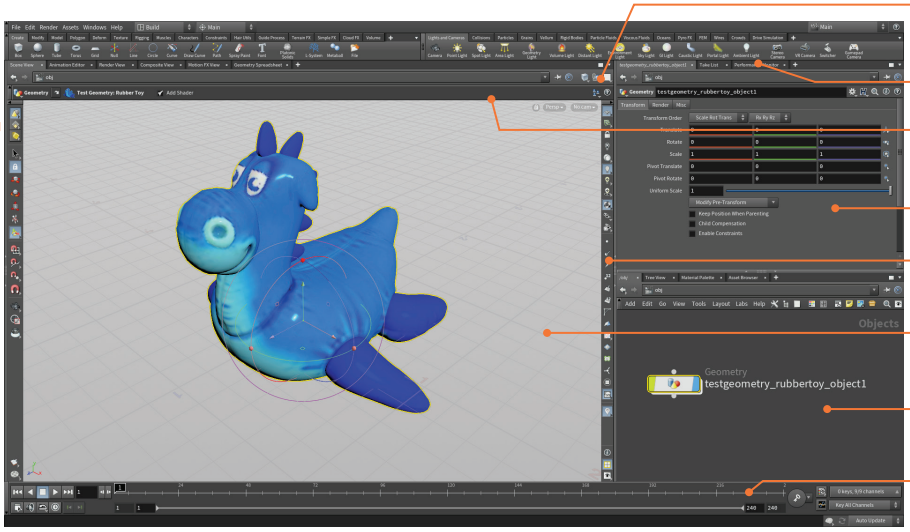
Select Tools - Select, Secure Selection

Transform Tools - Move, Rotate, Scale or Pose or the Handle tool for node-specific controls.

Snapping Tools - Turn on Grid, Primitive, Point or Multi-snapping.

Viewing - Use the View tool to Tumble, Pan and Dolly or Render Region to render in the Scene View. Click and hold to change to 2D Pan and Zoom.

Output Tools - You can render or flipbook your scene with these tools.



3D VIEW TOOLS

Here are some of the hotkey combinations available while viewing. You can skip **spacebar/alt** if you are actually in the **View** tool:

- **Tumble** Spacebar or Alt[Opt] - Left Mouse Button [LMB]
- **Pan** Spacebar or Alt[Opt] - Middle Mouse Button [MMB]
- **Dolly** Spacebar or Alt[Opt] - Right Mouse Button [RMB]

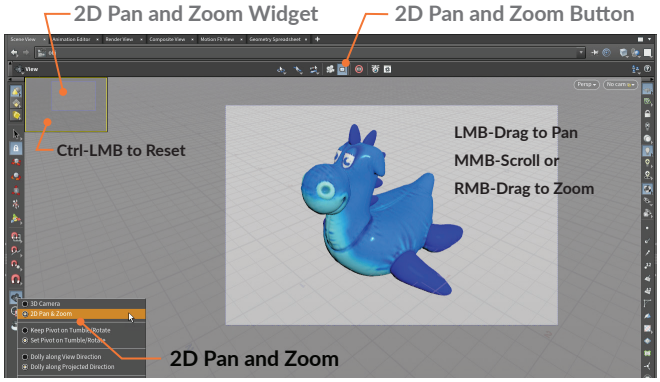
You can find the **View** tool in the toolbar. When you use the **spacebar** or **alt** keys, you temporarily evoke the view tool. This can be quite useful when you are selecting or manipulating in a view and need to quickly change your point of view.

If you want to focus on viewing then you can press **Escape** to go to the **View** tool. Here are some other hotkeys, you can use to get your bearings:

- **Home Construction Plane** Spacebar + H
- **Frame All** Spacebar + A
- **Home Selected** Spacebar + G

2D PAN AND ZOOM

You can click on the **2D Pan and Zoom** tool in the **Operation Control** bar to change your view in 2D without altering your 3D camera position. The widget at the top left lets you click to pan and zoom or to reset the view with **Ctrl - LMB** click. This is a great tool for working with locked off cameras.



Viewport Display Menus - These tabs let you create and organize multiple panes at the same time.

Pane Tabs - These tabs let you create and organize multiple panes at the same time.

Operation Controls - Use the Handle tool with this bar to access parameters from your selected node.

Parameter Pane - This pane lets you set values, add expressions and keyframe selected nodes.

Display Options Bar - These toggles let you control scene display options such as normals, point numbers or lighting.

Scene View - Visualize your work and use handles to manipulate objects interactively in your scene.

Network Pane - View and manage networks of nodes to work with the underlying structure of your scene.

Playbar - Set the current time and edit keyframes on selected nodes. You can also use the Playbar to copy and paste keyframes.

FIRST PERSON CAMERA

While in the **View** tool, you can turn on a first person fly through mode similar to those used in videos games.

- **Toggle First person On/Off** M
- **Dolly In/Out** W/S
- **Pan Right/Left** A/D
- **Look Around** LMB

VIEWPORT DISPLAY MENUS

Change how objects appear and views are organized using the menus in the top right of the Scene view or the **V** radial menu.

Shading Menu - Choose from options such as wireframe, flat shaded, smooth shaded or smooth wire shaded.

Object Display Menu - As you dive into networks, this menu sets whether you hide, view or see geometry ghosted.

Views Menu - This menu lets you split your scene view into various views such as perspective or orthographic views.

DISPLAY OPTIONS BAR

At the right side of the scene view, the display bar gives you access to options for viewport display. Here are a few examples.

Reference Plane/Ortho Grid - Turn on and off a grid that can be used for reference and for grid snapping.

Construction Plane - Turn on and off a construction plane which is used to define where you place objects or points.

Lock Camera- Lock the current camera to the view so that view changes modify the camera transform values.

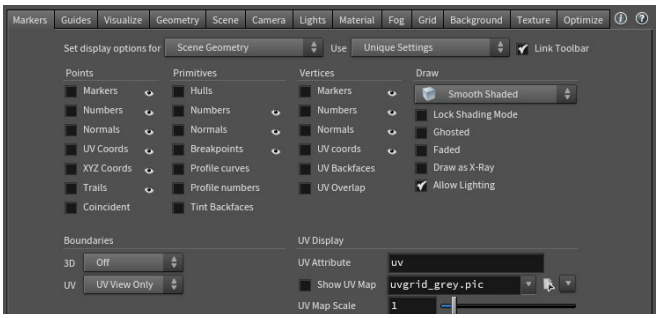
High Quality Lighting with Shadows- Set the best quality of viewport rendering.

Display Primitive Normals - Show the normals belonging to all primitives in the scene to determine their direction.

DISPLAY OPTIONS

The Scene and Network views each have Display option panels that you can access by clicking on the icon at the bottom of the Display options bar or using the following hotkey:

Display Options D



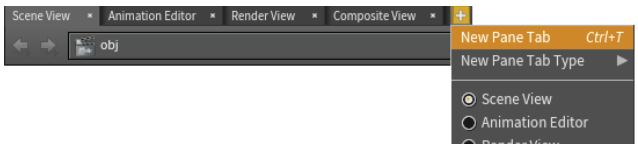
Panes & Desktops

The Houdini workspace is broken up into Panes which offer unique ways of organizing your scene data. You can work interactively in a 3D view or analyze attribute values in a spreadsheet. It is important to learn how these different UI elements can be used to get your work done.

PANES AND PANE TABS

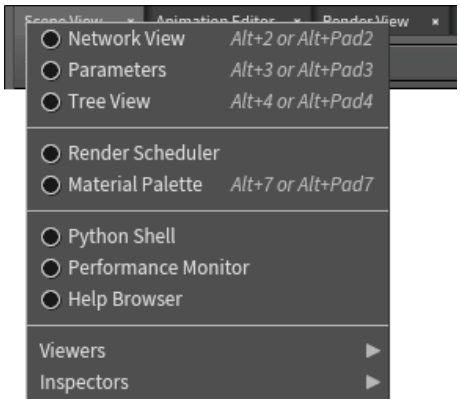
The Houdini workspace is divided into panes so that you can set up and explore your scene. Pane tabs let you overlap several panes within the same zone to keep them handy but not visible by default

You can access a pane tab by clicking on it in the workspace. You can close it by clicking on the x. The **+ menu** can be used to change the pane you are looking at or to add new panes.



PANE TYPES

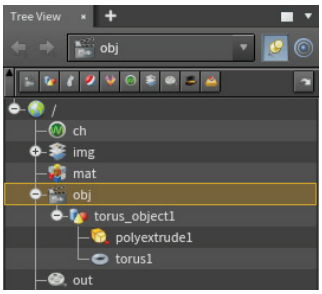
RMB-click on the pane tab to change its type. There are many pane types to choose from. Here are some of them which have hotkeys. Some of the other ones are listed here. Refer to the documentation to find learn more about all of the others.



Network View [Alt-2] - This view lets you see the nodes and networks and connect, rewire, and reorganize them to suit your needs.

Parameters [Alt-3] - Set values on parameters, add expressions and control the properties of your nodes.

Tree View [Alt-4] - This is a hierarchical view of the nodes. This can be a great way to understand how scene hierarchies work.



Viewers > Scene View [Alt-1] - Work interactively in 3D space. This type of view can be set up with one or more viewports. You can have more than one scene view panels open at the same time to look at your scene from different points of view.

Composite View [Alt-0] - View images and composites created using Compositing [COP] nodes.

Viewers > Motion FX View [Alt-^] - This lets you view motion created using Houdini's channel operator [CHOP] nodes.

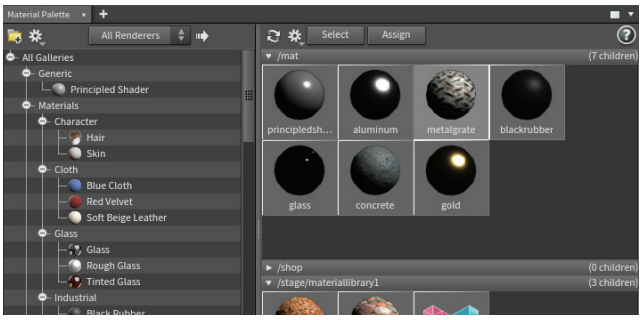
Solaris > Scene Graph- This pane shows you the USD scene graph when working with Solaris [LOP] nodes.

Solaris > Render Gallery - This pane lets you save test renders then revert back to the settings of each image after reviewing them all.

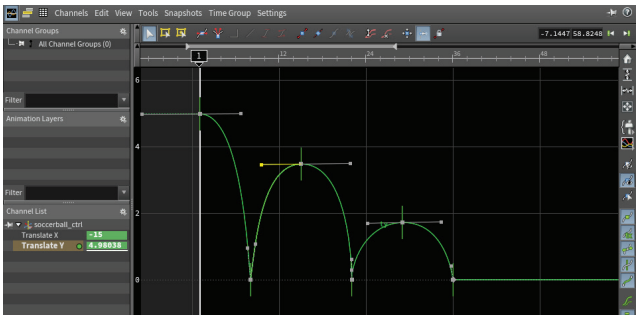
Solaris > Light Linker - Connect lights and objects.

Render Scheduler - This panel shows you Renderings that are completed and in progress. You can pause and kill renderings here.

Material Palette [Alt-7] - This palette lets you see all the materials in your scene and select and assign them to objects and geometry.



Animation > Animation Editor [Alt-6]- Manage keyframes and animation curves. The editor also has **Table** and **Dope** sheet views.



Animation > Channel List - Create channel groups and manage the scoped channels as you animate in Houdini.

Animation > Autorigs- This pane gives you access to tools for building your own rig out of modules for biped, quadruped and facial rigs.

Animation > Character Picker - You can use this pane to make it easier to select parts of a character rig.

Inspectors > Geometry Spreadsheet [Alt-8]- A view of the attribute values you have on your geometry. This could include UVs, normals, or custom values you have set yourself.

Inspectors > Data Tree- This view gives you access to a light bank, Material Stylesheet and Object appearance list.

Mantra Rendering > Render View [Alt-9] - Start interactive Mantra renderings, that will update when you change something in your scene.

Mantra Rendering > Take List- This list lets you explore different "takes" by making changes to specific parameters. You can then manage the takes to focus on your preferred creative choices.

TOPS > Task Graph Table - The Task Graph Table shows the metadata of all work items in the graph, or all work items in a particular node.

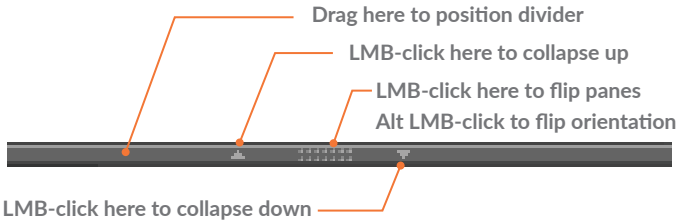
Misc > Orbolt Asset Browser - This browser lets you access assets from Orbolt.com. To use this pane you need to log in with an orbolt.com account.

Misc > Textport - You can use this pane to type commands.

Misc > Python Shell -You can type Python commands using this pane.

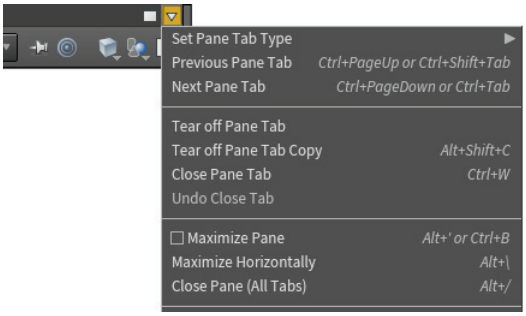
ORGANIZING AND COLLAPSING PANES

Both panes and tool bars can be collapsed and expanded by clicking on the arrows found in their UI. Whole panes can be collapsed to the left or right and you can flip the contents using the center grip. These options let you focus on certain panes by hiding others using a single click of the mouse.



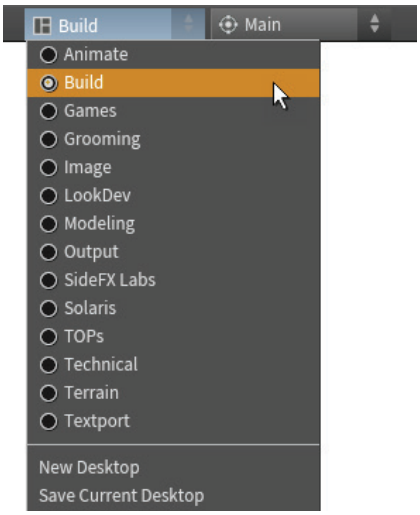
PANE MENU

At the top left of each pane is a button for maximizing and minimizing the pane and an arrow which gives you access to the Pane menu. This menu lets you tear off the pane or a copy of the pane, close, or split your pane. You also have options for determining the UI of each pane.



DESKTOPS

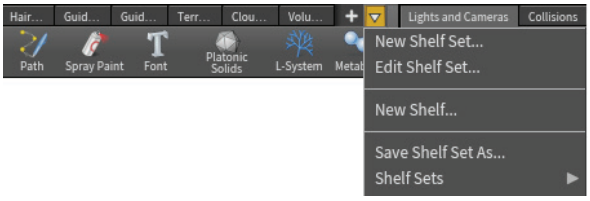
As you open up tabs, add dividers and organize your pane tabs, you start to set up your own workspace. To save any layout, go to the **Desktop** menu (**Windows > Desktop** on OSX) where you can access saved desktops, save your own and manage them as you work. When you save a desktop, it will save the Pane layout, Radial menus and visible Shelf sets.



When you save your scene, it remembers which **Desktop** you are looking at but not any changes to the pane layout while working. These changes will go away unless you explicitly save them to the desktop or create a new desktop.

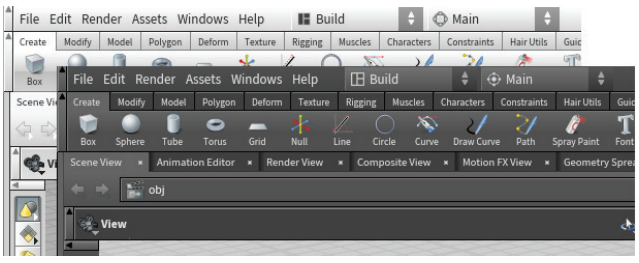
SHELVES AND SHELF SETS

To manage the shelves at the top of the workspace, access the menu found under the arrow icon. You can use this to work with Shelf sets. You can also bring up shelf sets that might be hidden in your desktop.



COLOR SETTINGS

You can customize the look of the Houdini UI by choosing a color scheme for your workspace. Select **Edit > Color Settings** to bring up the option window then you can choose from the default **Houdini Dark** or **Houdini Light.** Click on the **Download** button to choose from a list of color schemes created by the Houdini community.



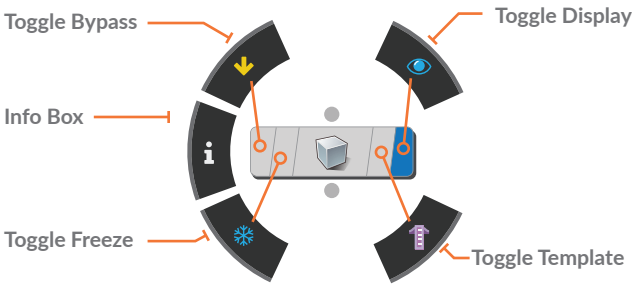
Nodes & Networks

With Houdini's node-based workflow at the heart of its procedural architecture, the ability to work directly with these nodes and networks becomes very important to using it effectively. While the idea of nodes might sound technical, they are actually quite artist friendly and easy to work with.

As you use tools in Houdini, nodes are created and wired with other nodes. The resulting networks offer a history of your actions while providing a simple way to make changes and refine your work. Learning how to work effectively with the node networks is an important part of working with Houdini.

NODE FLAGS

Each of the nodes have various flags which determine if it is displayed, locked or bypassed. You can evoke these by either clicking on the flag itself or using the radial node menu.



Display Flag [R] - This flag lets you choose the display output node for the network and is highlighted with a hollow ring. The **Render flag** [T] sets which node will be output for rendering and is highlighted with a solid circle. You can set this separately from the Display flag by **Ctrl-clicking** on the Display flag.

Template Flag [E] - This flag displays the node in grey and can be used for reference or snapping.

Freeze Flag - This caches at the locked node and all nodes earlier in the chain are ignored when the network is cooked.

Bypass Flag [B] - This flag lets you ignore the node when the network is cooked.

CONNECTING AND DISCONNECTING NODES

When you work in the Viewport, nodes are often placed and wired together automatically. When you want to reconfigure how a network is set up, you will need to connect and disconnect nodes by hand.

Here are some ways to interact with nodes and connections in the Network pane:

- **Connect Node** — LMB-drag from output to input
- **Connect Nodes** — J drag across the nodes
- **Insert New Node** — RMB on an output or connector
- **Insert Node** — LMB drag and drop onto connector wire
- **Disconnect from Wires** — LMB select then Jiggle node(s)
- **Cut Wire** — Y drag across connector wire
- **Move node** — LMB-drag
- **Copy selected nodes** — Alt + LMB-drag
- **Reference Copy** — Alt + Shift + Ctrl + LMB-drag

Dot nodes can be used to organize your networks:

- **Add Dot** — Alt + LMB wire
- **Pin/Unpin Dot** — Alt + LMB dot

NODE GALLERIES

The galleries offer quick access to nodes that you want to add to your network directly. The galleries contain those nodes used the most in day-to-day work while the **tab** key gives you access to all the available nodes.

You can create your own galleries using the **Windows > Gallery Manager** and you can add items to your galleries by **RMB-clicking** on a node then choosing **Save to Gallery...**

Nodes saved in the **Mat** network will also be available in the **Material Palette** as long as they are given the proper keywords such as *Mantra* for Mantra materials.

NETWORK VIEW

Network Path - The path leading to the current network level. You can also use this bar to navigate to other networks.

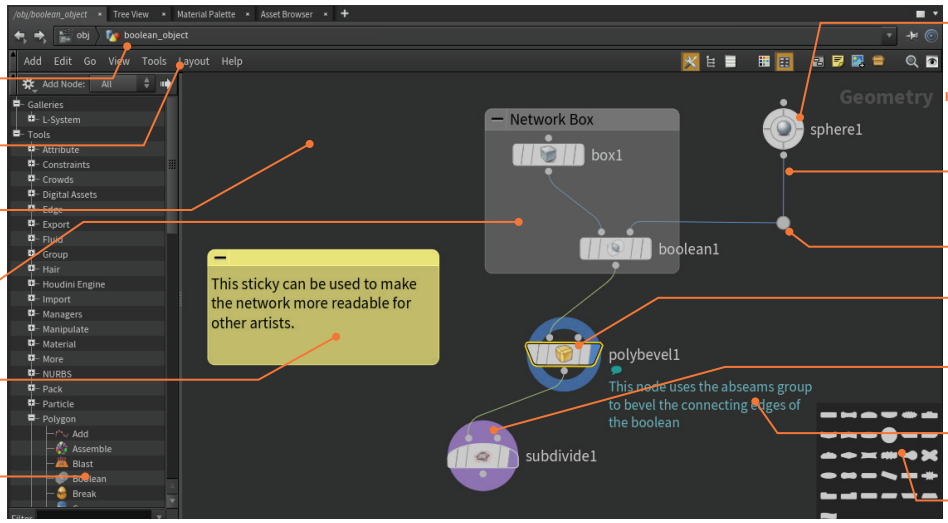
Pane menu - These menus and icons are for organizing your network.

Network Background - Use the Pane menu to add an image or set up a grid to help you organize your nodes.

Network Box - Group related nodes then quickly collapse and expand them.

Sticky - Add notes to help other artists read your network or to offer ideas for their networks.

Node Gallery - Drag nodes from here to your network. Use the filter at the bottom to find the node you need.



NETWORK TYPES

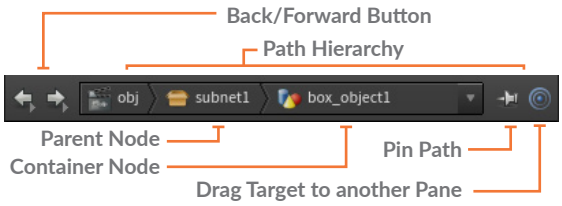
Houdini includes different kinds of nodes which each work in their own context. The network type is labelled in the top right corner of the network view. Nodes from each type can connect to other networks. While the different types of nodes are similar in how they are wired together, they each have unique capabilities:

Scene	Objects	OBJ
Geometry	Surface Operators	SOP
Solaris	Lighting/Layout Operators	LOP
Materials	VEX Builder	MAT
Motion FX	Channel Operators	CHOP
VEX	VEX Builder	VOP
Outputs	Render Operators	ROP
Tasks	Task Operators	TOP
Dynamics	Dynamic Operators	DOP
Compositing	Compositing Operators	COP/IMG

As you work with Houdini, you will begin to learn how to use this "secret" language to talk about the node types and how they apply to working procedurally.

NETWORK PATHS

Nodes are organized hierarchically with some nodes nested in other nodes known as network managers or subnetworks. To help you manage these hierarchies, a browser-like path is available at the top of most panes.



Use this path to navigate up and down the hierarchy or to other networks. By default, the path changes as you make selections in the scene view although you can **Pin** down a path to keep it focused. You can also drag the **Target** icon to a pinned pane to sync up their paths.

NAVIGATING NETWORKS

To jump between network types there are a number of different approaches you can take. Some of these happen naturally as you work with objects in the scene view and others offer shortcuts which allow you to work more quickly.

Selection Modes - As you select in the scene view, the network editor jumps to the location of the selection. Different selection modes will in turn take you to different network types as you make a selection.

Network path - You can **LMB-click** on a parent node to navigate back up the path or **LMB-click** on the container node to access parallel nodes or dive into the contents of other container nodes.

Radial menu - **Press n** to get a radial menu that lets you navigate up, down and to different network types.

Hotkeys - These help you navigate up and down as you work with a selected object.

Dive in	I
Jump up	U
Toggle Objects/Geometry	F8
Previous or Next Network	Alt + ← or Alt + →

Quick Marks- These let you quickly set and return to network locations. You can use them as-needed and then override them or forget about them. They are not saved with the scene file.

Set a Quickmark	Ctrl + 1, 2, 3, 4 or 5
Return to a Quickmark	1, 2, 3, 4 or 5
Go Back to Previous View	`

SELECT AND VIEW HOTKEYS

In the network pane, you will need to pan and zoom around to work with the complete network. Here are the key combinations for these actions.

Pan	MMB
Zoom	RMB
Select Nodes	LMB
Add to Selection	Shift + LMB
Remove from Selection	Ctrl + LMB

Node - This represents an operation that contributes to the final output of the network.

Network Type - Shows which network type you are working in.

Connector - The connecting lines show how your nodes are linked together and how the data is moving through the network.

Dot - You can add dot connectors to make it easier to organize your nodes.

Display Ring - This small circle shows which node is displayed in the Scene view.

Render Ring - This large circle indicates a render node even if another node is displayed.

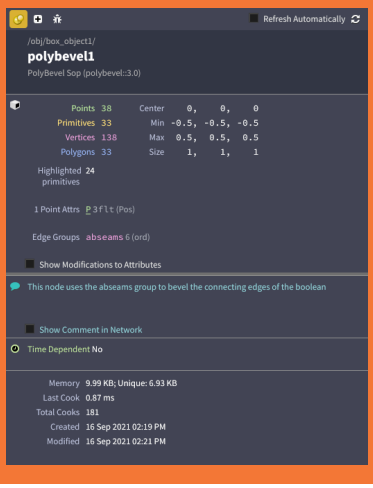
Comments - Node comments can be displayed to help other artists understand your thinking.

Palettes - Buttons on the menu display Palettes that let you set the color and shape of nodes.

LEARN ABOUT YOUR NODE

Bring up the **Info Box** using either the **Radial Menu** or **MMB-press** on the node. This panel gives you info about the node's contents, groups, attributes and other important facts. This panel also highlights any errors that are interfering with your workflow.

This panel will close automatically but you can click on the **Pin** icon to keep it visible as you work. You can add comments and display them in the Network view using this panel.



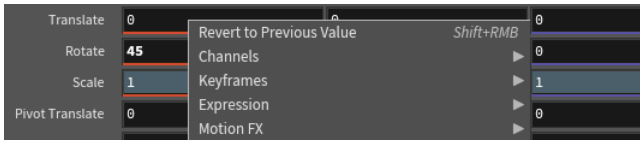
Parameters, Channels & Attributes

All of the nodes in Houdini are driven by parameters, channels and attributes to help you achieve the results that you want. The terminology used in Houdini may differ from other 3D applications therefore it is a good idea to take a moment to understand them in a Houdini context.

PARAMETERS

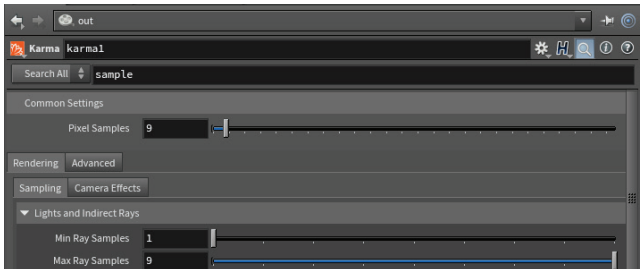
Parameters refer to the values, sliders, buttons and checkboxes found on Houdini nodes. These are sometimes referred to as attributes in other applications but Houdini uses attributes in a different way.

You can change parameter values in the Parameter pane or using handles in the viewport. There is a RMB menu on each parameter that gives you a number of important options such as copying and pasting and reverting to defaults.



SEARCHING PARAMETERS

A node might have a large number of parameters and sorting through them all can take time. If you click on the magnifying glass in the top right, you get a search bar that lets you filter the parameters based on name and content. You can find parameters using expressions, overrides and even a raw value.



CHANNELS | KEYFRAMES

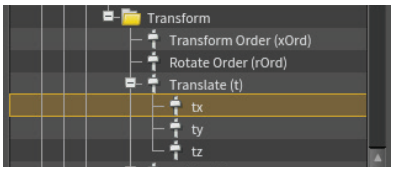
You can set a keyframe on a parameter by pressing the **Alt** key and **LMB-clicking** on the name or value field. Once you have set a keyframe, the parameter's field changes color and you have created an animated channel. There will now be keyframes associated with the parameter which you can access in the Animation editor.

CHANNELS | EXPRESSIONS

Instead of a raw value, you can add expressions into the parameter using either *hScript* or *Python*. There is a menu in the top right of the Parameter pane for choosing which language you want to use. You can press **Ctrl-E** to bring up an expression editor with a number of scripting tools to make it easier to work.

REFERENCE SCENE DATA

You can also **RMB-click** on a parameter and choose **Reference > Scene Data** to bring up a window for choosing specifically what you want to link to. Once you have made a choice from any node in your scene, to create a channel reference. This method lets you create references without worrying about the exact syntax needed to write the proper expression.



PARAMETER PANE

Navigation Bar- This bar lets you see where the node is located in the scene hierarchy.

Node Type and Name - Here you can see the node type and set its name. Clicking on the icon gives you a menu for working with the node.

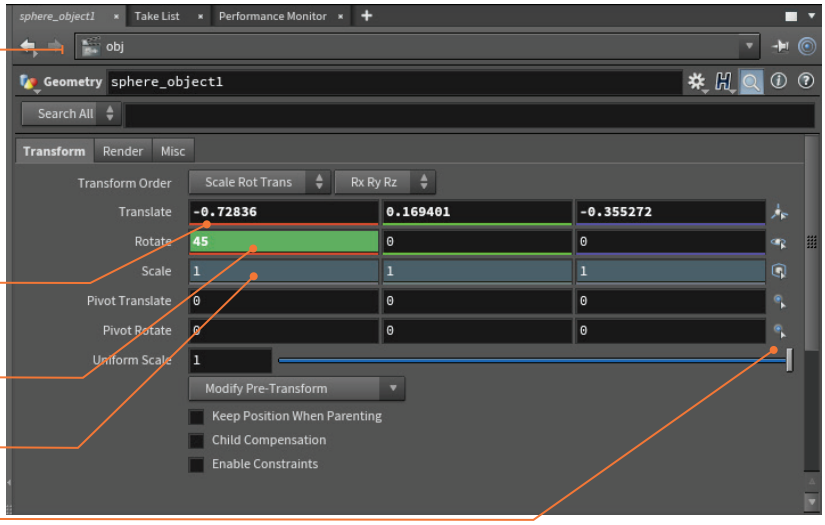
Search Bar - Click on the magnifying glass icon to search parameters by name or by content.

Changed Parameter - When a parameter has been changed from its defaults then its value is bolded. The folder tab name is also bolded.

Animated Parameter - When you have keyframed a parameter it is highlighted in green.

Locked Parameter - You can RMB-click on a parameter to lock and unlock it. It will be highlighted in grey

Select to Match - These icons let you match these parameter values to other objects.

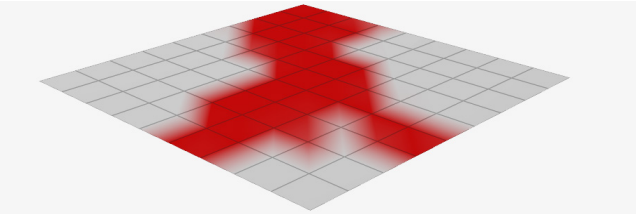


CUSTOM PARAMETERS

If you click on the Gear icon in the top right of the Parameter pane, you can choose **Edit Parameter Interface**. Here you can add custom parameters which can then be linked to other parts of your node network.

ATTRIBUTES

Attributes let you attach data to your geometry that can be used by nodes down the chain to complete an operation. A fuel attribute can drive a Pyro FX simulation or a UV attribute sets up texturing. Some attributes are created by Houdini nodes or you can create custom attributes.

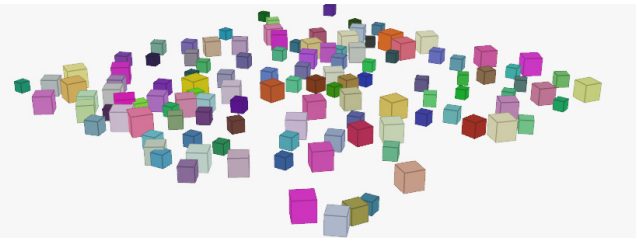


Class - Attributes can belong to Points, Primitives, Details and Vertices. This will affect how they get used down the chain.

Type - You can set up float, integer or string attribute types amongst others.

ATTRIBUTE RANDOMIZE

Attribute Randomize lets you create an attribute and immediately randomize its values. For instance here you can see the Color, rotation and Scale of these boxes being randomized.

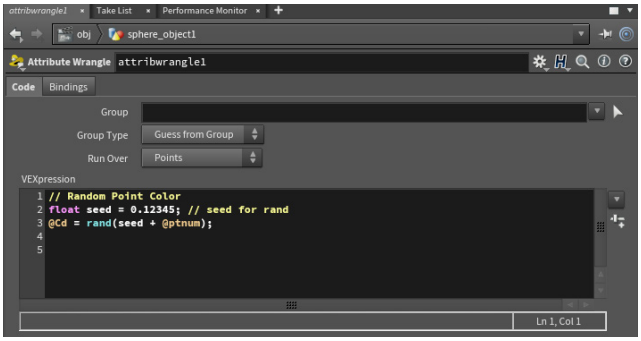


ATTRIBUTE TRANSFER

Within a node chain, attributes are attached to geometry then used by other nodes. You can also pass attributes to other pieces of geometry using **Attribute transfer**. Here the sphere is passing color attributes to the boxes based on a defined threshold value.

ATTRIBUTE WRANGLE

Houdini has a wide variety of nodes that let you create and work with attributes. You can also use the **Attribute Wrangle** node to use a script-based approach to this work. For a lot of Technical Directors this may be the most comfortable way to work.



For artists, working with nodes will make it easier to deal with this kind of information. A lot of Houdini's power is found in the proper use of Attributes and you will eventually need to learn about them.

GEOMETRY SPREADSHEET

You can view all attribute values, even invisible ones using the geometry spreadsheet.

Navigation Bar- This bar lets you see where the node is located in the scene hierarchy.

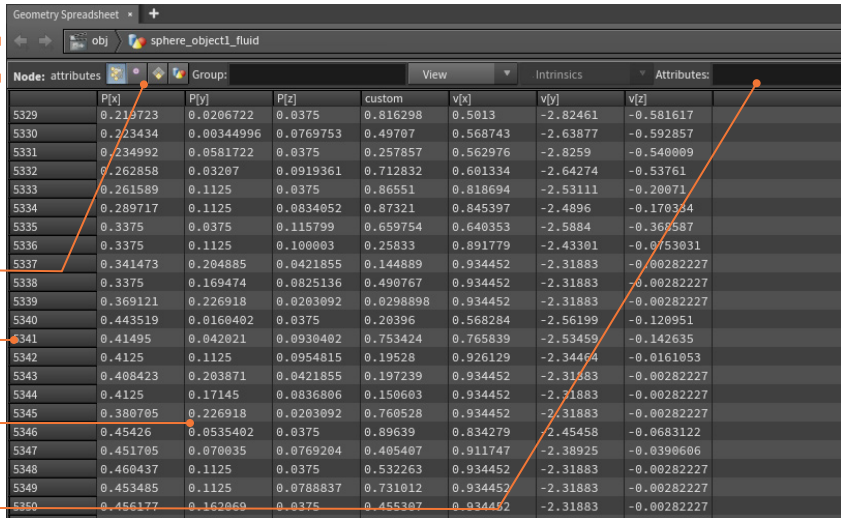
Node Name - This shows you which node is currently selected and which node is generating these attribute values.

Attribute Class Buttons - Use these buttons to filter which kind of attribute you are looking at.

Point Number - Here are the geometry point numbers to help you determine where the attribute is on your model.

Attribute Values - These are the values at this point in the node network chain.

Filter - You can type parameter names here to filter the list when you are working with lots of parameters.



	P[x]	P[y]	P[z]	custom	v[x]	v[y]	v[z]
5329	0.219723	0.0206722	0.0375	0.816298	0.5013	-2.82461	-0.581617
5330	0.223434	0.00344996	0.0769753	0.49707	0.568743	-2.63877	-0.592857
5331	0.234992	0.0581722	0.0375	0.257857	0.562976	-2.8259	-0.540009
5332	0.262858	0.03207	0.0919361	0.712832	0.601334	-2.64274	-0.53761
5333	0.261589	0.1125	0.0375	0.86551	0.818694	-2.53111	-0.20071
5334	0.289717	0.1125	0.0834052	0.87321	0.845397	-2.4896	-0.170334
5335	0.3375	0.0375	0.115799	0.659754	0.640353	-2.5884	-0.368587
5336	0.3375	0.1125	0.100003	0.25833	0.891779	-2.43301	-0.0753031
5337	0.341473	0.204885	0.0421855	0.144889	0.934452	-2.31883	-0.00282227
5338	0.3375	0.169474	0.0825136	0.490767	0.934452	-2.31883	-0.00282227
5339	0.369121	0.226918	0.0203092	0.0298898	0.934452	-2.31883	-0.00282227
5340	0.443519	0.0160402	0.0375	0.20396	0.568284	-2.56199	-0.120951
5341	0.41495	0.042021	0.0930402	0.753424	0.765839	-2.53459	-0.142635
5342	0.4125	0.1125	0.0954815	0.19528	0.926129	-2.34464	-0.0161053
5343	0.408423	0.203871	0.0421855	0.197239	0.934452	-2.31883	-0.00282227
5344	0.4125	0.17145	0.0836806	0.150603	0.934452	-2.31883	-0.00282227
5345	0.380705	0.226918	0.0203092	0.760528	0.934452	-2.31883	-0.00282227
5346	0.45426	0.0535402	0.0375	0.89639	0.834279	-2.45458	-0.0683122
5347	0.451705	0.070035	0.0769204	0.405407	0.911747	-2.38925	-0.0390606
5348	0.460437	0.1125	0.0375	0.532263	0.934452	-2.31883	-0.00282227
5349	0.453485	0.1125	0.0788837	0.731012	0.934452	-2.31883	-0.00282227
5350	0.456177	0.162069	0.0375	0.455307	0.934452	-2.31883	-0.00282227

Selecting Geometry

Working in Houdini involves the selection and manipulation of many different elements. There are a number of tools and options available to help you work efficiently with objects and geometry components such as points, edges and primitives.

SELECT TOOL

The **Select** tool lets you focus on making selections therefore it doesn't have any manipulation handles.

- **Select Tool** — Tap S

When working with tools such as **Move** or **Rotate** and **Secure Selection** is on, you need to invoke the **Select** tool to make a selection. Toggle **Secure Selection** to **Off** to select freely.

- **Evoke the Select tool while in other Tool** — Press and Hold S
- **Toggle Secure Selection** — ~

SELECTION TYPES

There are different shortkeys for adding, subtracting or toggling your selection as well as for selecting all or none. These techniques play an important part in this workflow.

- **Select** — LMB
- **Add to Selection** — Shift + LMB
- **Remove from Selection** — Ctrl [Cmd] + LMB
- **Toggle Selection** — Ctrl [Cmd] + Shift + LMB

- **Select All** — A [Object Level] / N [Geometry Level]
- **Select None** — N [Object Level] / Shift + N [Geometry Level]

SELECTION TECHNIQUES

In the viewport, you can choose from four different selection types that offer different ways of accessing geometry.

- **Box Select** — F2
- **Lasso Select** — F3
- **Brush Select** — F4
- **Laser Select** — F5

There are also some selection filters that let you focus on visible geometry or select groups. You have a wide range of selection options to help make this easier as you work.

- **Select Visible Geometry Only** — Shift + V
- **Select Fully Contained Geometry Only** — Shift + C
- **Select Groups or Connected Geometry** — 9
- **Select Whole Geometry** — Choose in Operation Control bar
- **Select by Normals** — Choose in Operation Control bar

SELECTION MODES

Selection modes, give you access to objects and components. They also let you easily jump from object level to geometry level using a the buttons in the toolbar or the hotkey.

Objects - The object network level is where you work with an object's transforms. In any tool other than the **View** tool, the following hotkey will bring you back to the Object level:

- **Objects** — 1

Geometry - You can use any of the following hotkeys, when not in the **View** tool, to jump into the geometry level with the chosen components available for selection.

- **Points** — 2
- **Edges** — 3
- **Primitives [Faces]** — 4
- **Vertices** — 5

TWEAK MODE

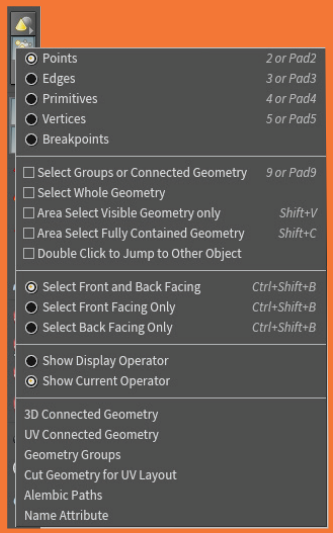
Only one geometry selection modes can be active at a time. If you are working with an **Edit** node then **Tweak Mode** lets you choose any combination of points, edges and primitives.

SELECT MODE MENUS

Each of the selection modes comes with options which let you alter how you interact with your scene. You can access these options by either **LMB** or **RMB-clicking** on each mode's icon.

While working with components, this menu lets you choose to show **Display** or **Current** Operators. These same options are available at the top of the Scene view when working with the **Edit** node.

This menu is different at the object level where it includes some filters for different kind of objects as well as options for more easily selecting materials, constraints and Digital Assets.



SELECTION OPTIONS

Edit | Components You can choose which components you want to work with from this collection of buttons. Here edge selection has been chosen.

Select Tool - The **Select** tool lets you make the selection. To access it press the **S** hotkey.

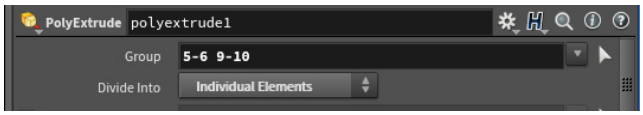
Secure Selection - This locks your selection when using other tools. To invoke the **Select** tool with it on, press and hold on the **S** hotkey.

Selection Types - You can use this top bar to change your type of selection. You can choose from Box, Lasso, Brush or Laser. There are also some filter options.

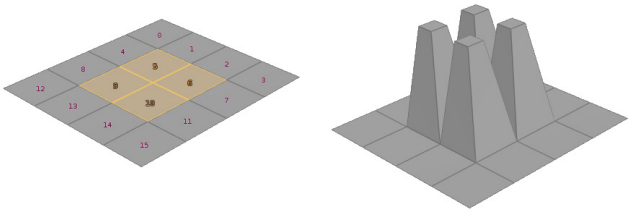
Edge Loop - To select an edge loop you can **double-click** while selecting edges. To select partial loops, select one edge then press **A** and then an end edge. This works with points and primitives. You can also select point loops or primitive loops using the same technique.

HOW SELECTIONS ARE USED BY TOOLS

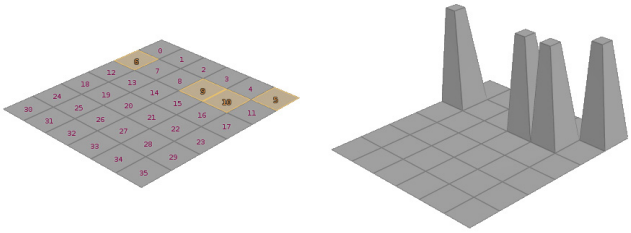
When you make a selection in the viewport then use a tool, a node is created and the selected points, primitives or edges are listed in the node's **Group** parameter.



For example here we see primitives **5, 6, 9 and 10** are being used by a *polyextrude* node. You can see them listed in the Group node and then used to extrude the faces.



If you were to change the topology of the incoming geometry node then there might be more or less faces and the extrusions will have moved to a different location. This may not be what you want and you may need to reselect faces.



To do this, you can select *polyextrude*, press **Enter** to go to the **Handle** tool and press ` to go into re-select mode. Select new primitives then press **Enter** and your new selection will be used in the **Group** parameter.

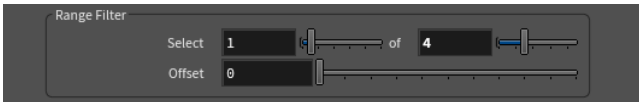
SELECT ALL AND THE GROUP FIELD

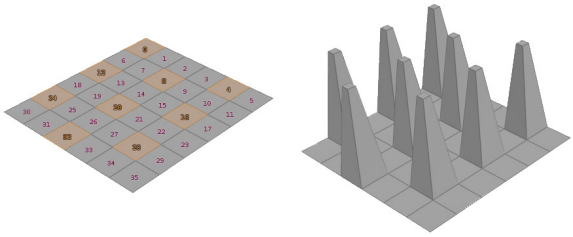
To select all of the primitives on incoming geometry, leave the **Group** parameter blank. If the topology of the incoming geometry changes, everything will be operated on by the node.

Using **Select All [N]** in the viewport will usually ensure that this field remains blank when a tool is used. With some tools, the Group field will show all the selected parts and you would have to clear the field manually to make it blank.

THE GROUP NODES

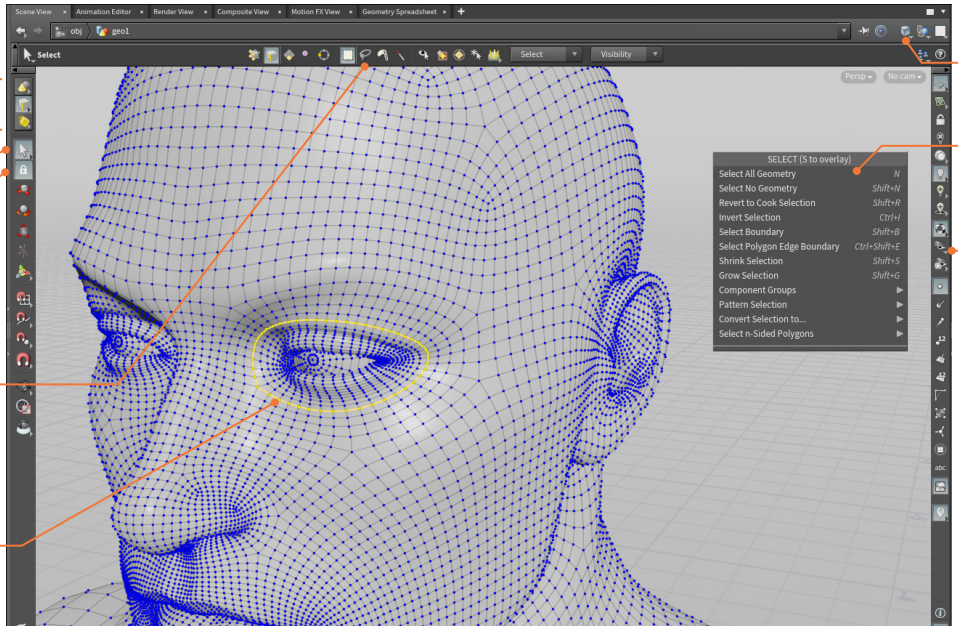
Group nodes let you refer to a defined selection of points, vertices, polygons, or edges by **name**. You can define a group interactively, by selecting components in the viewer, or mathematically, using ranges or an expression. The group name can then be assigned to the **Group** parameter instead of using point or primitive numbers.





Here are some **Group** nodes for you to choose from:

- **Group Create** - Use interactive selection, a bounding box, face normal direction or edge angles to populate the group.
- **Group by Range** - This lets you choose a range and a simple pattern to populate the group.
- **Group Expression** - With this node you can use a vex expression to define the membership of the group.
- **Group Paint** - This node lets you use an interactive paint interface to choose the geometry for the group.



Shading Options - The shading options determine what you see in the Scene view. In this case, we are using Smooth Wired Shading.

RMB Menu - While in the Select tool, this menu gives you access to selection options such as inverted selections, boundaries or growing and shrinking your selection.

Display Filter - This filter lets you turn off things you don't need such as bones, null objects, lights or cameras to let you focus on the work at hand.

Display Options - While selection modes will show you edges or points to help you select, they will not be visible when using other tools. Use these options to keep them visible even when not using a specific modeling tool.

Transform & Edit

From basic transformation tools for objects, to the pose tool for animation rigs and the edit node for reshaping geometry, there are a number of different tools that let you use interactive handles in the viewport. In Houdini, these handles are tied closely to the node you are working with.

TRANSFORM TOOLS

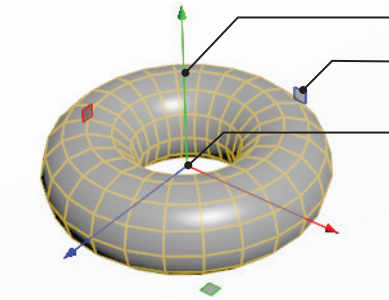
The transform tools give you handles that you can use to manipulate objects or reshape geometry. When you transform objects, the parameters at the object level are updated to reflect your changes.

- **Move** T
- **Rotate** R
- **Scale** E
- **Pose** Ctrl-R
- **Handle** Enter

The **Handle** tool gives you access to handles that are specific to your selected node. While using these tools, you can re-select by **pressing and holding S**, making a new selection then **releasing S** and continuing to transform.

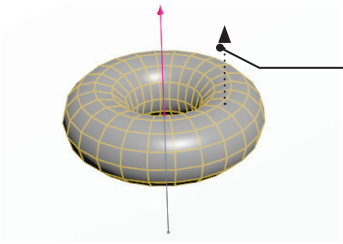
TRANSFORM HANDLES

When you are working with a **Move** handle you can work with a single axis, two axes or move along the camera plane using the center. **Rotate** and **Scale** handles offer similar controls.



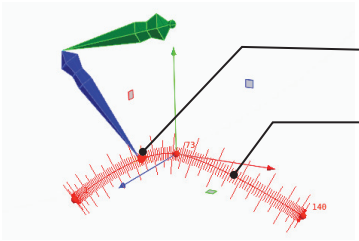
MMB TRANSLATION

If you don't want to click directly on handles, you can use the **Middle Mouse Button** in open space combined with a drag to move it along the construction plane. To change this to translate along nearest axes go to **Edit > Preferences > Handles** and set Translate Handle to **Map Drag to Axis.**



POSE TOOL

When you are animating, you can use the **Pose** tool to work with bones and to display motion path handles that reveal the motion of an object. You can then use tangent handles and keyframe points to modify the motion in the viewport.



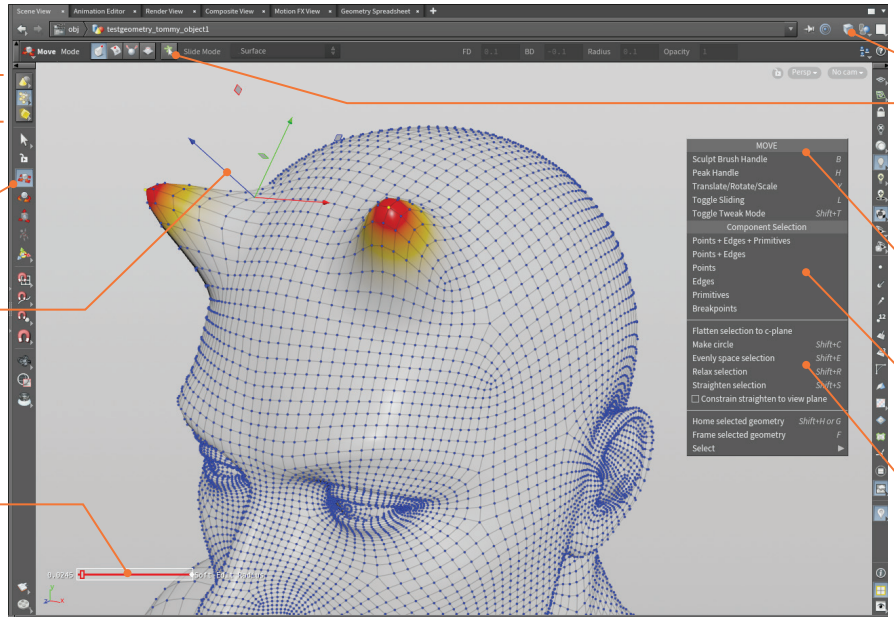
EDITING GEOMETRY

Edit | Components You can choose the components you want to edit using these buttons. The **Points** option has been chosen here.

Move Tool - The **Move** tool lets you translate the selection using the Scene View handle.

Move Handle - This handle lets you move along one axis using the lines or two axes using the square dots. **RMB-click** on the handle to access the handle options.

Soft Edit Radius - When moving points on a surface, you can use this radius value to create a soft falloff as you move the points. The Soft Edit Radius doesn't work with primitives or edges.



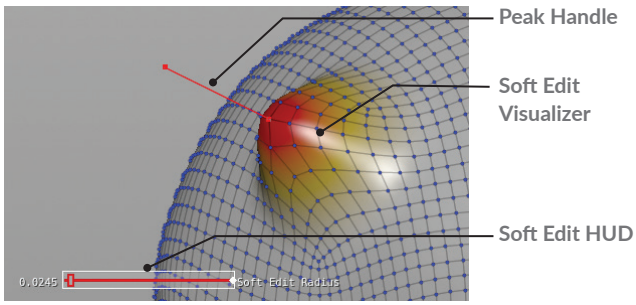
EDIT NODE

If you try to move geometry components then an **Edit node** is placed down to accept your transformations. In addition to transforming the geometry, you can slide on surface, work perpendicular to the normals or sculpt the surface.

- **Edit** T/R/E
- **Slide on Surface** L
- **Peak** H
- **Sculpt** B

SOFT FALLOFF

When you are transforming points, you can use the **Soft Edit Radius** to create a falloff. There is a visualizer that is evoked to show you where the falloff is occurring on the surface.



EDIT OPTIONS

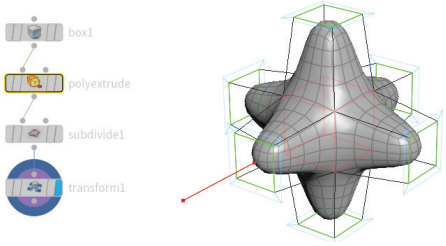
If you **RMB-click** while in the Edit node, you can access options for transforming your selection. You can make a circle or straighten the selection. These options work with points and edges but not always with primitives.

- **Make Circle** Shift-C
- **Evenly Space Selection** Shift-E
- **Relax Selection** Shift-R
- **Straighten Selection** Shift-S

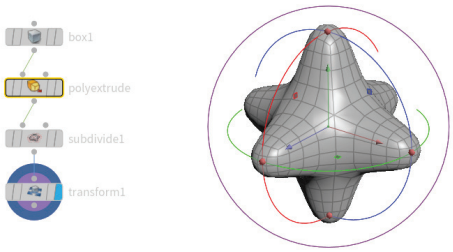
HANDLE TOOL

After using a shelf tool, you often find yourself in the **Handle** tool. Or you can select a node in the network and press the **Enter** key in the Scene view to go into the **Handle** tool. This brings up a handle which focuses on the specific parameters for the selected node such as the **distance** parameter on a *polyextrude* node.

Show Current Operator - By default when you select a node other than the display node, it becomes the current node and you get a wireframe display of the geometry. You can then use the handle to manipulate this intermediate node while evaluating the results on the shaded surface.



Show Display Operator - Another option is to always show the Display Operator. In this case, selecting a node in the chain does not show the wireframe and the handles stay focused on the display node.



You can change parameters in the parameter pane for the current node but the handles will continue to work with the parameters on the display node.

Shading Options - The shading options determine what you see in the Scene view. In this case, we are using Smooth Wired Shading.

Sloppy Selection - Three of the component buttons can be selected at the same time if the Edit node is using Sloppy selection which makes all of them available for a more fluid selection process.

RMB Menu - This menu gives you access to Edit tool options such as which type of edit you want to focus on. This information is also available on the top bar of the Scene view.

Component Selection - You can also select the component type using this menu. This offers you the same options you would find on the main toolbar.

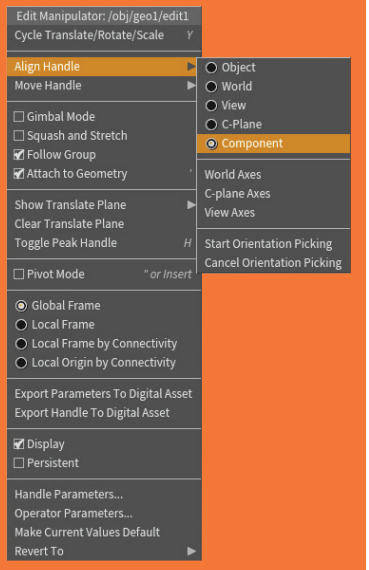
Edit Options - You can use this menu to edit the components using operations such as **Make circles** and **Straighten** the selection.

HANDLE OPTIONS

All handles have a menu that can be accessed by **RMB-clicking** on any part of the handle.

This menu gives you options for aligning the handle, detaching it from the parameters of the node, pivot mode and more. You can use these options to customize how the handle works.

You can also keyframe parameters on a handle and promote all parts of the handle to a Digital Asset. By promoting the parameter the handle will be accessible at the asset level.



Modeling Tools

Houdini has a lot of tools for creating, shaping and deforming geometry to achieve a desired look. Here are just a few of the many tools you will use on a regular basis when building models in the geometry, or SOP, context of Houdini.

CREATION

To start creating geometry, you can start with some basic shapes or draw a curve. In each case you get an object with a geometry/SOP node inside with the tool's name. You can access these on the **Create** shelf or in a radial menu.

Primitives - Houdini includes Box, Sphere, Tube and Torus primitive shapes along with a variety of Platonic solids.

Grid - The grid tool offers a great starting point for a wide range of models. You can set its shape and size at the geometry level.

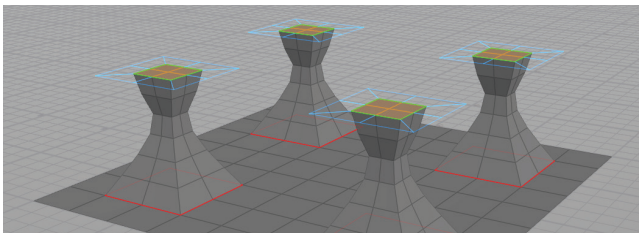
Curve - Draw a curve by laying down control points then create a Bezier, NURBS or poly curve.

POLYGON MODELING

Polygons are one of the most popular geometry types especially in video game projects where they are mandatory. Houdini has a comprehensive set of poly modeling tools which you can use to develop your models.

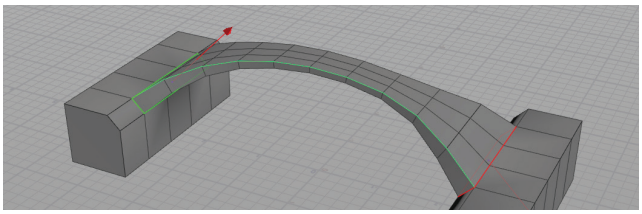
PolyDraw - This tool lets you interactively draw a polygonal mesh on the construction plane or by snapping to existing geometry.

PolyExtrude - Push or pull on one or more polygons to reshape the geometry. Control the extrusion profile to get a wide variety of shapes.



PolyBevel - Bevel selected edges to create chamfered or rounded bevels. You can often use the output group from a previous node such as Polyextrude or Boolean to automatically find the right edges.

PolyBridge - Connect two sets of polygons with control over the shape of the bridge.



PolySplit/Edge Loop/Knife - These tools let you split polygons to add more detail to your model.

PolyExpand 2D - Take curves and edges that sit on a 2D plane and creates geometry based on a desired offset value.

PolyReduce - Create different levels of detail by reducing the number of polygons while preserving quads and UVs.

PointWeld- Interactively snaps groups of points to another target point, and merges them.

UTILITY NODES

Because of Houdini's procedural nature, modeling actions like copy, clip and mirror create nodes in your network. This can makes it easier to go back and make changes later on.

Clip - Cut your model based on a clipping plane. You can set the direction of the clip and whether you keep one half, the other or both.

Mirror - This tool flips the geometry based on a clipping plane. There is an option to fusing the points after mirroring.

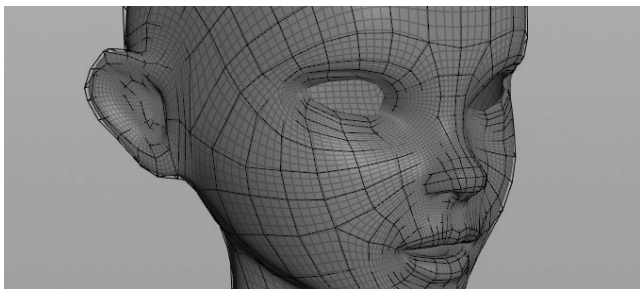
Copy and Transform- This node will let you create multiple copies based on transformation values.

Blast - This node lets you delete polygons from your model. You can choose to remove or keep the selected polygons. If you press the **Delete** key when points or a polygons are selected they will be blasted.

Dissolve - This tool lets you remove edges without breaking up the surrounding geometry. Pressing the **Delete** key when an edge is selected will dissolve it.

SUBDIVISION SURFACE MODELING

In Houdini, you can model with polygons then **display** and **render** them as subdivision surfaces using options found on the **Render** tab on the object's parameter pane. You can also create a **Subdivide** node at the geometry level to add polygons to give you a more detailed topology to work with.



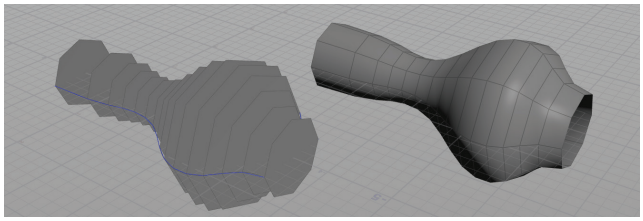
SURFACING TOOLS

There are tools in Houdini that will take profile curves and build surfaces. Those input curves can be either bezier, polygonal or NURBS or a mixture of them.

Revolve- Create geometry by revolving a profile curve around an axis. There is a handle available for tweaking the results.

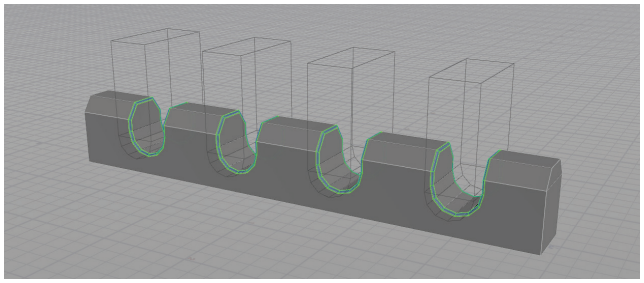
Skin - Take a series of profile curves and turn them into a surface.

Rails - Copy one or more profile curves along two or more rail curves, then Skin the results to get a surface.



BOOLEANS

Subtract, **Union** or **Intersect** geometry using the Boolean tool. This node can handle very complex topologies and can be used to break up a surface for destruction using rigid body dynamics. This often creates more realistic results compared to the Voronoi-based **Shatter** node.

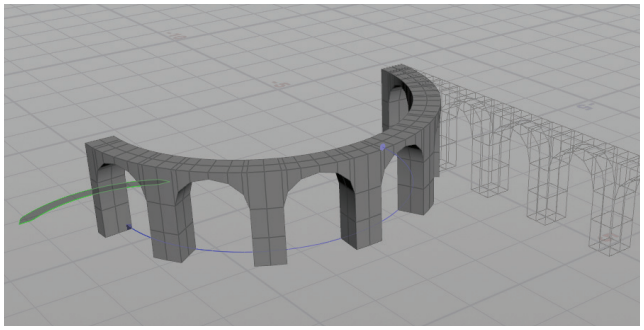


The Boolean tool can create output groups that you can use to feed other nodes such as the **Polybevel** node. This way any updates you make to the boolean will update properly when it feeds into the second node.

DEFORM TOOLS

While you can shape your geometry by editing points directly, there are times that you need a more generalized approach. The following nodes provide options for shaping your geometry procedurally.

Bend - This node lets you set a capture range and direction then bend, twist, taper and squish the encompassed geometry.



Lattice - This builds a lattice around your geometry then lets you edit points on the cage to reshape it. You can also use a custom cage.

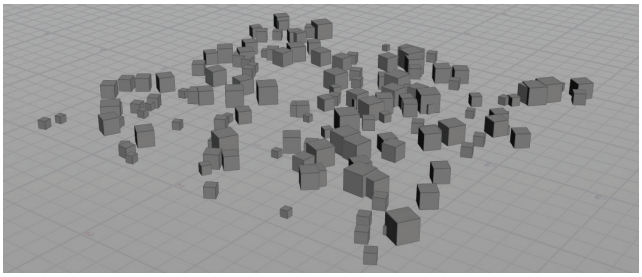
Mountain - Apply a noise function to deform the surface to create a random result. The points are actually being moved with this node.

Ripple - This node creates a ripple shape in your geometry.

Waves - This node adds noise functions to create a wave-like pattern that animates over time. Perfect for creating realistic oceans.

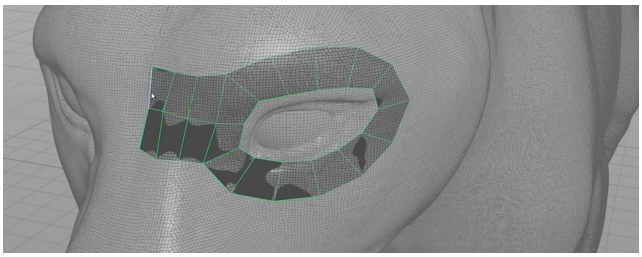
COPY TO POINTS + SCATTER & ALIGN

A typical Houdini workflow is to **Scatter & Align** points on a surface then **Copy to Points**. Attributes for scaling and rotating the objects can then be applied to create a more organic result. This is often used to create landscapes with trees and rocks.



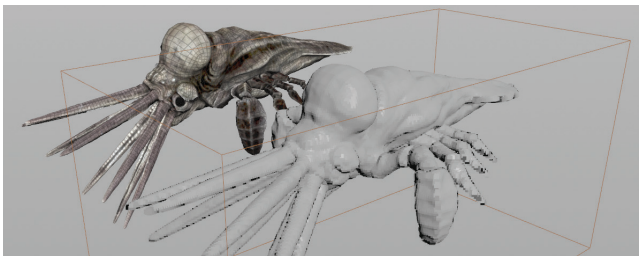
TOPOBUILD

Houdini has a **Topobuild** node that lets you draw polygons directly onto high-resolution geometry that you either scanned or created in an application such as Pixologic's Z-Brush. You can create a cleaner topology for animation then bake the details from the original model into a normal map.



VOLUMES

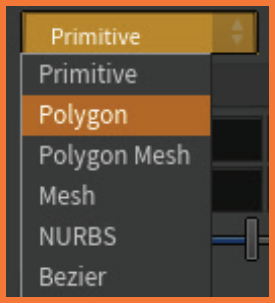
Volumes allow you to store values for voxel, or three dimensional pixels, in a space. These are often used to support collisions when using dynamic tools or to create clouds. They can also be used for modeling to combine multiple shapes into a single volume which you then convert back to a surface.



GEOMETRY TYPES

Houdini supports a number of different geometry types including **Primitives**, **Polygons**, **NURBS** and **Beziers**. You can **Convert** back and forth between them and you can have more than one geometry type merged together in a single object.

Polygons models can be set up to display and render as **Subdivision Surfaces** using **PIXAR's OpenSubdiv** standard. Both Subdivisions and NURBS will render in Karma and Mantra as perfectly smooth without relying on any tessellation settings.

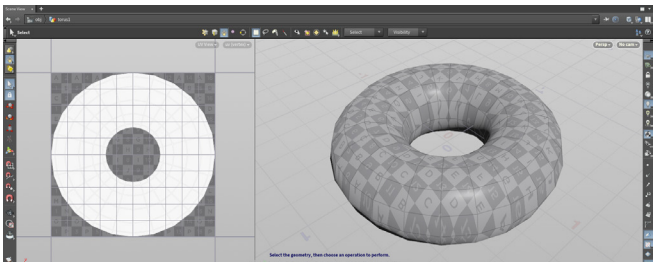


UVs & Textures

To make 2D maps fit properly onto 3D objects, UV coordinates are needed to define a flattened view of your geometry. When you first create geometry in Houdini it will not have any UVs. Even primitive objects do not have any built-in UVs. This means that you will need to add them at the geometry level using one or more SOP nodes.

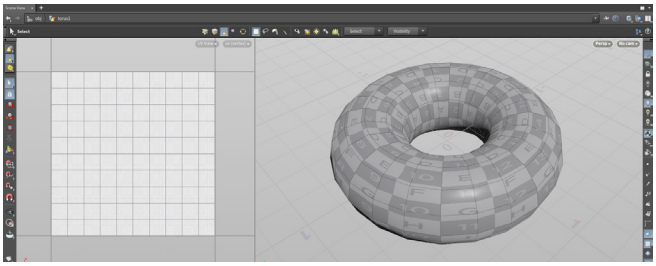
UV TEXTURE DISPLAY

Since geometry in Houdini does not have UV texture attributes set up by default, you need to add them using UV tools. Once you have UVs set up, you will see a texture grid on your geometry because **Show UV Texture** is on in the **Display Options** bar. You can toggle it off if you don't want to see the UV texture or change it to a color texture.



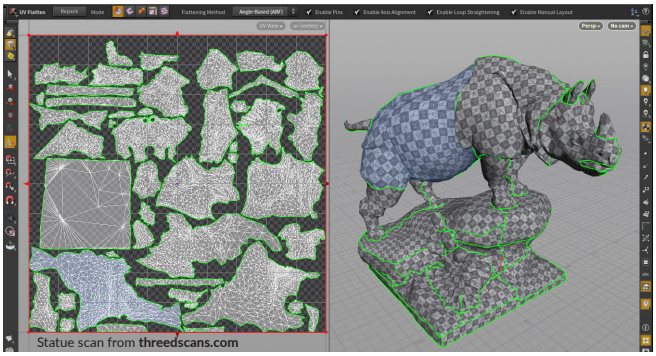
UV PROJECT

This node let you assign UVs using one of several projection techniques. Once you choose your projection type you can **initialize** the projection to match your object. This may invert the UVs and you will need to set a **rotate x** value of **-90** instead of 90. Above you can see an **Orthographic** projection and below you can see a **Toroidal** projection.



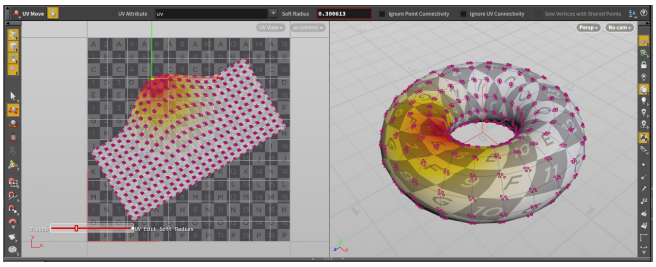
UV FLATTEN

UV Flatten unwraps your geometry based on predefined boundaries created using selected edges or edge groups. The results can then be tweaked by pinning points in the **UV view** and adjusting the islands to get the look you want.



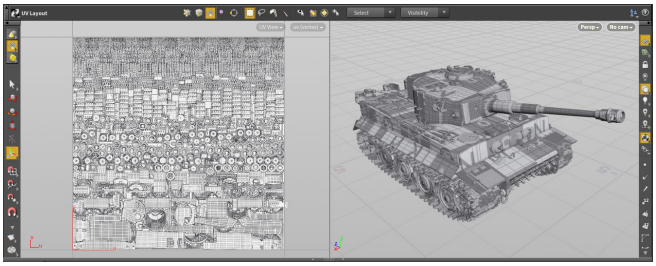
UV EDIT & DISTORTION

To edit individual vertices or groups of vertices you will use either the **UVedit** or **UVtransform** nodes. The UVedit node lets you perform many edits using a single node while UVtransform allows one edit per node which can provide a more procedural result. You can use **Display > UV Distortion** from the **UV Viewport menu** to see whether you are editing too much.



UV LAYOUT

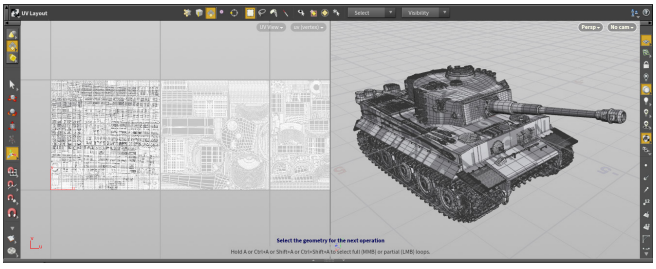
UV layout will let you create UV islands and pack them into UV space as efficiently as possible. This lets you maximize how much of a texture is being used on your geometry which is important when optimizing for both rendering and gameplay.



You can use the region handle to put a the UV layout into a particular part of your UV space. Subsequent layouts can then work around this layout using the **Pack Islands in Cavities** option.

UDIMS

In addition to working with a Single UV tile, you can use UDIMs to spread your UVs over many tiles. This technique lets you create more detailed texture maps because your UV islands are not packed in too tight. Properly numbered texture maps will then be assigned to the appropriate tile.



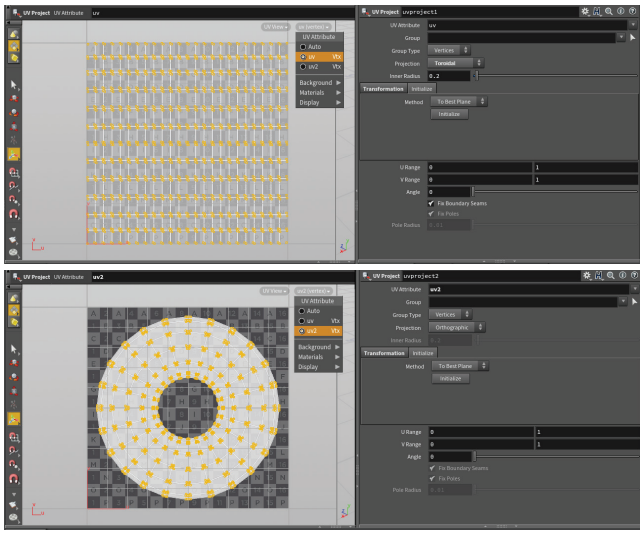
UV ATTRIBUTES

Earlier, you learned how attributes can be assigned to geometry that carry important information down the line. UVs are vertex attributes that let you wrap texture maps around your model and are also carried down your network.

These attributes are visualized in the UV viewport and analyzed in the geometry spreadsheet. These attributes work with various SOP nodes including the attribute wrangle node which lets advanced TDs manage their UVs using scripting.

UV SETS

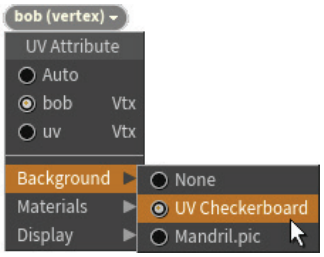
You can create more than one UV set on the same geometry. When you use the UV nodes, you can set the UV attribute. By default this is *uv* but you could create a *uv2* to create a second set. These different UV sets are used when you assign textures in VOPs so that different texture maps use different UV attributes.



In the two images shown here, the first uses a toroidal projection and is assigned to a *uv* **UV attribute** while the second uses a planar projection and is assigned to a *uv2* **UV attribute**. These UV attributes can have any name, for instance, it could be *bob* instead of *uv2*.

UV VIEWPORT MENU

The UV viewport menu lets you display UVs based on the UV attribute. You can also use this menu to figure out the background image which can either be the default UV grid or a texture map pulled from an assigned material.

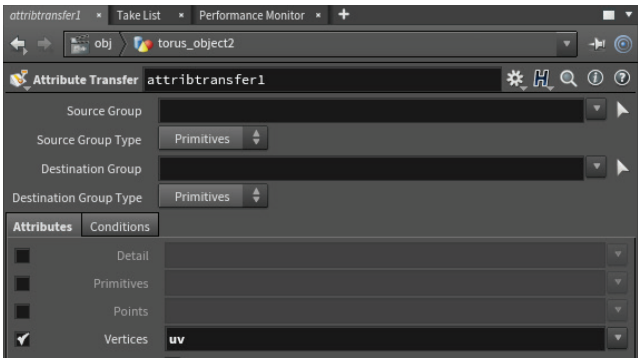


This menu also has display options for displaying **UV Overlap**, **UV Backfaces** and **UV Distortion**. These can be helpful when evaluating your UVs to decide if more tweaking is required.

ATTRIBUTE TRANSFER

One of the SOP nodes that can be used to manage attributes is **Attribute Transfer** which lets you take the UV attributes from one piece of geometry and transfer them based on proximity to another.

This can be useful when the topology of a model has changed but you want to preserve some of the work you did creating UVs for the original model.

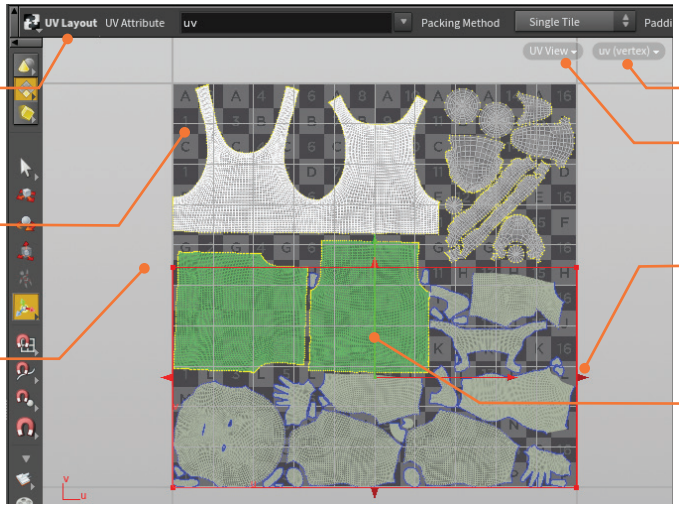


SCENE VIEW | UVS

Current Tool - At the top of the Scene View, you will find the selected node when the Handle tool is active.

Background - The background of the main tile can be set using options found in the UV menu.

Outside Main Tile - Polygons positioned in the area outside the main tile would overlap the same area on the main tile because the texture repeats unless you are using UDIMS which cover more than one tile.



UV menu - When you are in the UV view, this menu gives you a number of different options for working with UVs.

View menu - Use this menu to choose the UV view for this viewport.

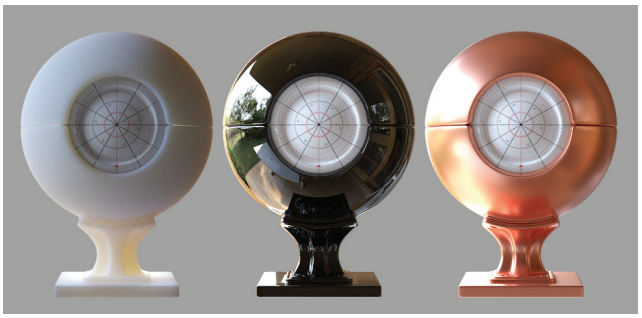
Layout Handle - This handle is part of the UV Layout node and lets you focus the UVs to a certain part of the tile.

Pack Around - The UV layout will pack your UVs around any existing geometry that has already had its UVs set.

LookDev: Shaders & Materials

To render objects in your scene, you must assign materials, also known as shaders, to your geometry. In Houdini, these materials and shaders are created in the mat/vex builder networks. The ability to build up materials using nodes is a powerful tool when defining the look of a shot.

Houdini organizes different types of nodes into network types and for materials you will use the `/mat` network type. This is where you can set up **VEX operators** for Karma and Mantra or **Material X** for Karma. Material X is an open standard, that originated at Lucasfilm,® which is used or the transfer of look-development content between applications and renderers.



THE MATERIAL PALETTE

You can add VEX-based materials using the **Material Palette** then assign them to objects by clicking and dragging. This pane has a workspace for managing the materials in your scene which is organized into tabs that represent different subnetworks such as **Material Library LOPS**. Use the **tab key** to add Material X shaders to material networks. Once they are in place they will show up in the Material Palette.

ASSIGNING MATERIALS IN LOPS

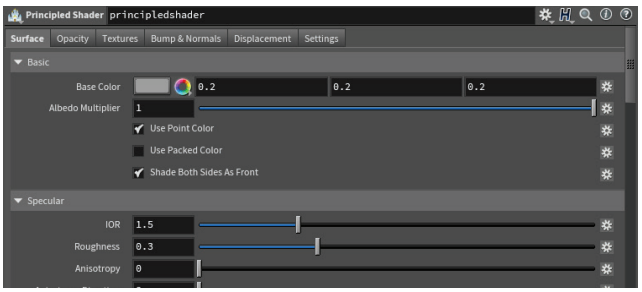
To assign a material in the Solaris LOP context, first create a **Material Library LOP** which contains a */mat* network. You can assign the material using this node or use an **Assign Material LOP** later in the chain. The **Material Palette** will let you drag from the gallery to the library or in LOPS you can use the **arrow button** to access a list of materials in your scene graph.

PRINCIPLED SHADER

In the Material Palette, you will find the principled shader, a material based on the Disney "principled" BRDF by Brent Burley. This shader is "principled" rather than physical in order to make it easier for artists to work with.

The **Principled Shader** has been pre-built to include the ability to assign textures directly to parameters such as base color, bumps, normals, displacements and more. The texture maps you assign will be displayed in the viewport and you can achieve a wide variety of looks right out of the box.

You can extend this material by feeding it with other VOPs but that isn't necessary in all cases. Many of the materials found in the gallery are variations on this shader.



PRINCIPLED SHADER CORE

The **Principled Shader Core** node sits inside the **Principled Shader** and contains the main features of the shading model but doesn't have all of the texturing features built-in.

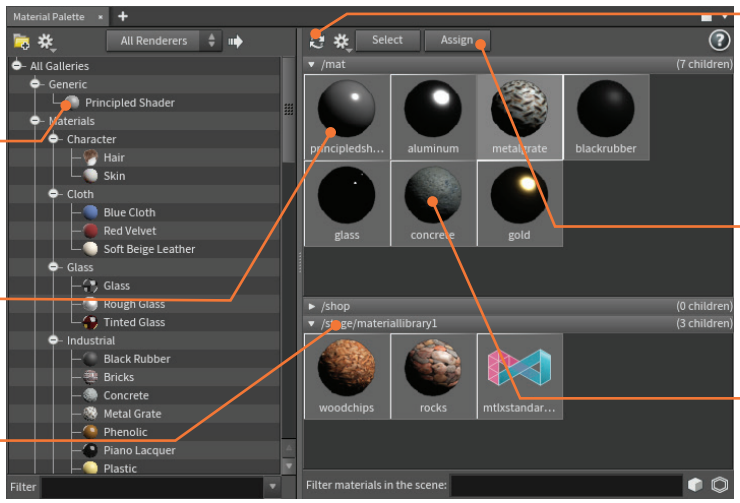
To build a robust shader from scratch using this node, you would need to add VOP nodes using Houdini's shader building tools. You can accomplish this by either wiring nodes together in the Node view or adding them using the Shader FX menus.

MATERIAL PALETTE

Materials in Gallery - The Materials listed here are saved to disk in a gallery file. You can drag these into the scene area found on the right or onto objects in the viewport.

Material in Scene - This is where you find materials that are part of your scene file. They can be assigned to objects by dragging from here into the viewport.

Material Library LOP - Materials set up in the LOP context can be placed into Material Libraries - you can drag materials into here from the gallery.



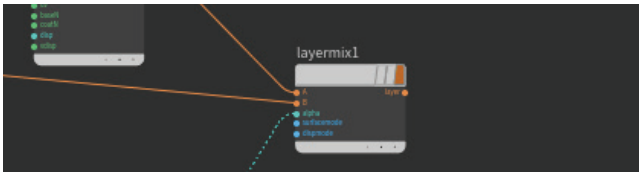
Update Material Icon - To update all material icons you can click on this button or RMB-click on material to do it one at a time.

Assign Material - If you select an object in the scene and the material in the palette then you can use this button to assign the material.

Double Click to Edit - If you double-click on any of these materials, you will jump to a Node View where you can make edits at the /mat level.

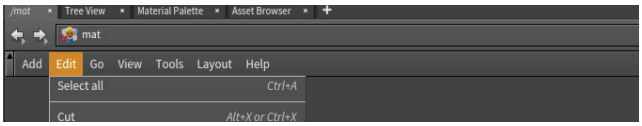
LAYERING MATERIALS

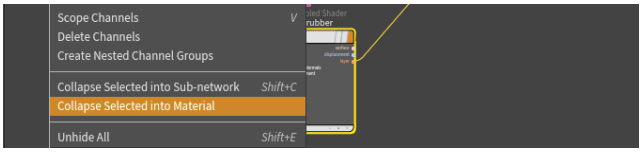
You can layer materials to create a unique look for your object. Using a **Layer Mix** node, you can combine two different materials into a single look. For instance, a shiny metal material and a matte rust material can be layered using this technique. You can then texture an alpha channel and you can choose to mix the surface, the displacement or both.



MATERIAL BUILDER

If you want to turn your layered nodes into a new material then you can select them and choose **Edit > Collapse Selected into Material**. This puts the nodes inside a **material builder** where you can continue to tweak it. At this level, there are *output* and *collect* nodes to make the network work efficiently.



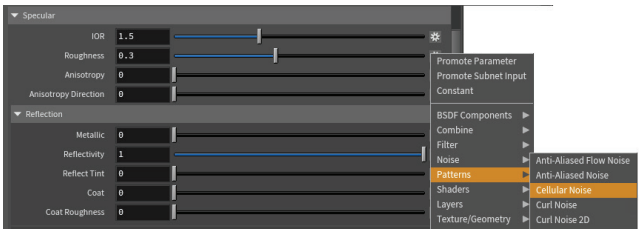


MATERIALS AS DIGITAL ASSETS

You can make materials even more efficient by saving them out as **Houdini Digital Assets**. In the **Asset Properties** pane, you can go to the **Save** tab and choose **Save Cached Code** so that the material is pre-compiled when you render with Mantra. You can also load texture maps into the digital asset then reference them from inside the asset file. Turning materials into HDA's can make it easier to share with your team.

SHADER FX MENU

While working with material VOPs, you can add nodes in the Network view and wire them together or you can use the **Shader FX Menu** by clicking on the icon at the far right of each parameter. This menu lets you focus on the parameter that you want to work with and create the node in context.



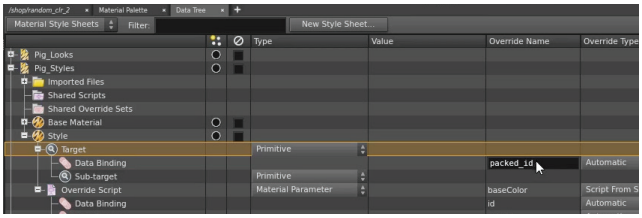
In the Parameter Pane, you can see what type of connection each parameter has by looking at the icon on the far right:

- No Connection
- Parameter Node
- Connected with other Nodes
- Hidden Connection

PROCEDURAL MATERIAL ASSIGNMENT

When working in production with lots of data, it is often necessary to take a procedural approach to assigning materials. With **Solaris** and **Karma,** you can accomplish this using nodes such as the **Assign Material LOP** or **Material Variation LOP.**

If you are working with **Mantra**, materials can be assigned to the object using the **Data Tree** panel which gives you access to **Material stylesheets.** Stylesheets let you use rules to assign materials and textures to large groups of objects.

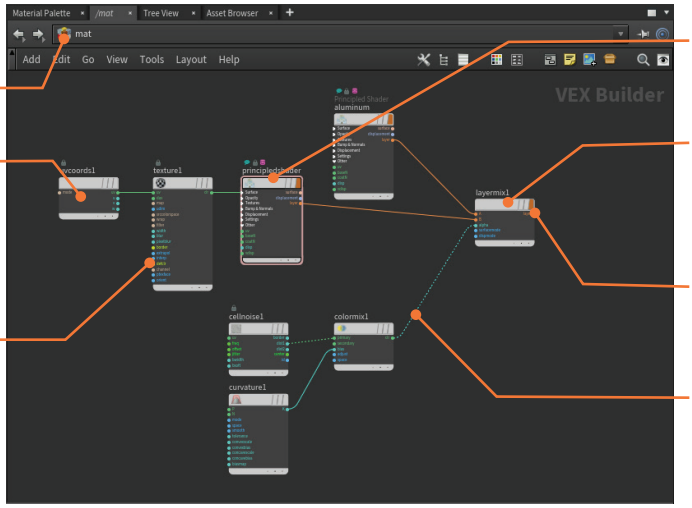


SHADER BUILDING

Node Path - This will confirm your current path and help you navigate into and out of the material network.

VOP Nodes - In the material context, you can start with material nodes then wire in VOP nodes to customize the texturing of the material. Once you are finished you can choose to collapse all of this back into a material builder node.

Node Connectors - You can add nodes to this area using the Shader FX menu which can be accessed by **MMB-clicking** on the dot. **RMB-clicking** on the dot gives you a full node menu.



Principled Material - This is a typical material which can be assigned on its own or fed into the Layer mix.

Layer Mix - The Layer output on the materials can be fed into this mix node. You can assign this to your geometry.

Material Flag- If you want the layer mix to show up in the material palette then check this flag.

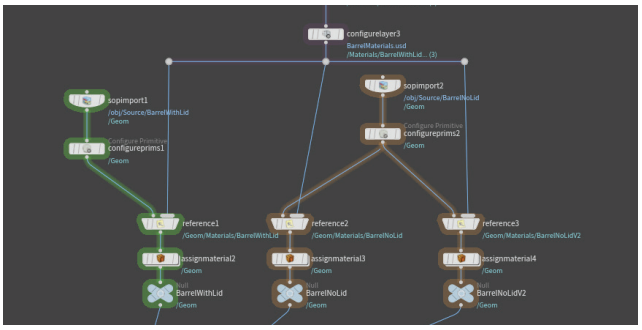
Alpha - Here the layer mix node's alpha is being fed by VOP nodes to create the alpha mask for the two layered materials.

Solaris: Layout

Solaris is the context in Houdini dedicated to Lookdev, Layout and Lighting that has USD at its core. Objects and geometry brought into this context become USD and you can use specialized nodes for positioning objects, instancing geometry and managing shot layout.

SOLARIS: LOPS

The Solaris environment can be found in the `/stage` network or by creating a LOP network. Here you will place nodes for bringing in geometry, assigning materials and adding lights and cameras. These nodes let you create sequences and shots using shared assets and procedural edit nodes to customize settings for each shot.



At its core, the Solaris environment converts everything into USD [**Universal Scene Description**] which is an open source initiative created by PIXAR. The Solaris/LOPS context allows you to work with USD natively using procedural nodes to manage USD concepts such as references, payloads, layers, collections, variants and level of detail.

LOP networks can then be rendered using either Karma or other Hydra-compatible renderers. Hydra is the technology that lets you render your USD to the viewport for interactive exploration or to disk for final renderings.

SCENE IMPORT: OBJECTS TO LOPS

For artists used to working at Houdini's Object level for layout and lighting, the **Scene Import LOP** node makes it easy to bring your geometry, lights and cameras into LOPS to render. If your goal is to create a controlled USD scene graph then you may want to take another approach but for quick access to Karma and other renderers the Scene Import node works fine.

PREPARING ASSETS FOR USD

Another approach is for you to configure props using either a **Component Builder LOP** network or the **USD Export SOP**. This would be geometry and materials set up on a prop-by-prop basis then exported as USD for use in the layout stage.

For some of the props, you will use **variants** to create different variations that can be chosen during layout. There are LOP nodes that let you set up this USD concept.

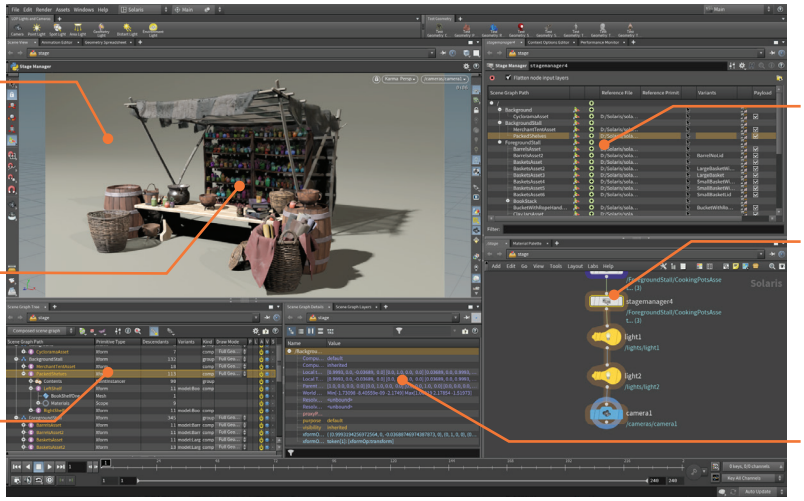
Once you have the props set up as USD, the files can then be brought into a larger layout scene file using **Sublayer** or **Reference LOPS** then **Edit LOPs** can be used to position the props. If the assets are prepped properly then they will come in with geometry, materials and variants set up and ready.

SOLARIS DESKTOP

Stage - The Stage view provides a place for viewing your layout and lighting and to interact with primitives, lights and cameras to set them up properly.

Viewport Render - On the stage, you can render using Houdini GL or Storm the USD GL solution. You can also render interactively to Karma and Hydra compliant 3rd party renderers.

Scene Graph - The USD structure offers a renderable scene graph that can be inspected using this panel.



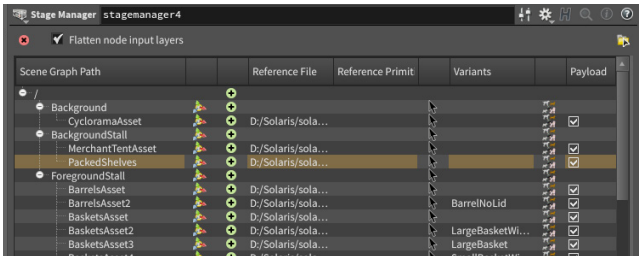
Stage Manager - This node lets you load USD files to be positioned on the stage.

LOP Nodes - All your actions in LOPS are accomplished using procedural nodes that make it easy to step back and make changes.

Scene Graph Details - As you select items in the Scene Graph you can check for details that inform you of its status in the pipeline.

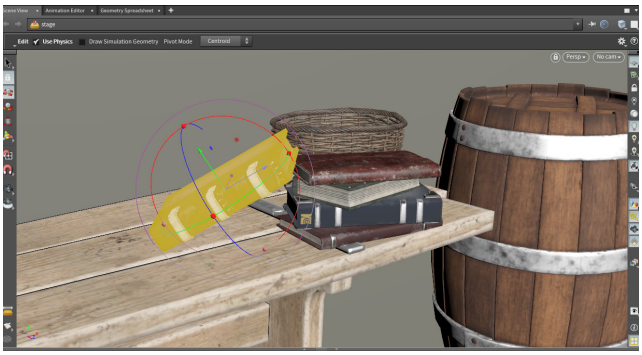
STAGE MANAGER

The Stage Manager is designed to be a one-stop node for referencing USD assets from disk, transforming them in 3d space, and adjusting your scene hierarchy. This does involve a flattening of the inputting layers which will block up-stream changes from coming through. This lack of flexibility is balanced by the rapid setup possibilities of this node.



PHYSICALLY-BASED EDITS

Once you have all of the props loaded onto the Stage, you can use the **Edit LOP** to begin moving them around. The edits become a separate non-destructive layer which keeps the referenced assets intact in case you need to step back.

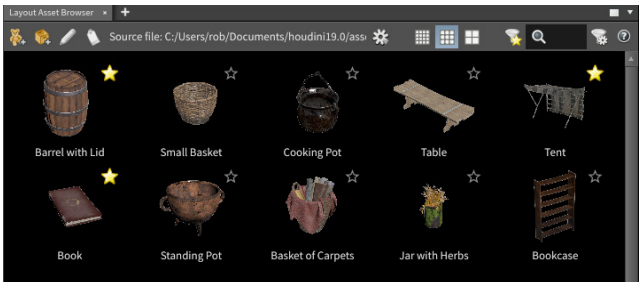


In the **Edit LOP**, there is a **Use Physics** option that lets you leverage the rigid body capabilities of Houdini to detect collisions and make it easier to position objects in a realistic manner. This lets the artist achieve a natural organic look while working interactively in the 3D view.

INSTANCING

There is an instancing solution in USD that can be utilized in LOPS. The Instance LOP lets you input one or more objects that will be distributed to points set up inside the LOP. These points can be created by importing geometry from your scene then scattering points on surfaces extracted from the model.

Materials can be assigned to the instances using a variety of methods including the **Material Variation LOP** which also offers per-geometry render properties.



You can also use the **Layout Asset Browser** and the **Layout LOP** to use a brush workflow to place, edit and transform instance points that reference the assets you choose in the browser.

EXPORTING USD

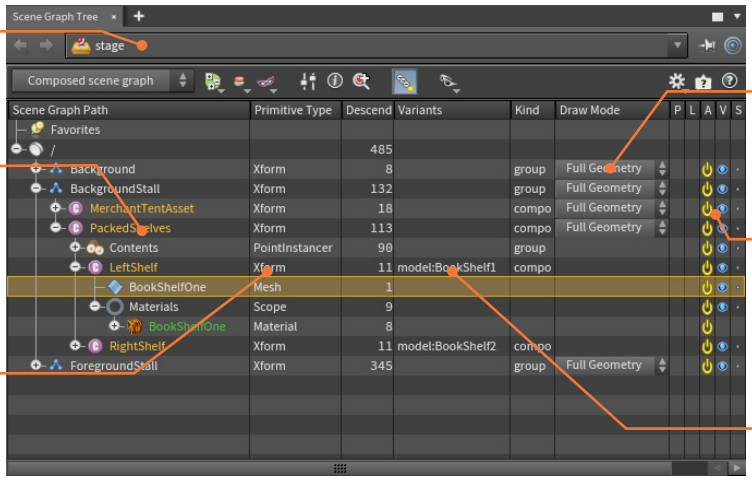
At various points in your LOP graph, you can inspect the USD code by RMB-clicking on the node and selecting **LOP Actions > Inspect Active Layer**. You can also inspect the **Flattened Stage**. When you export to USD, you can break out all the layers or save it as a single flattened graph.

SCENE GRAPH

Stage - This is the top level network context for creating LOPS. You can also do all of this work in LOP Networks.

Scene Graph Path - These represent the USD layers and sublayers that define the look of the stage. These are most likely different USD files on disk being referenced into a shot.

Primitive Type - Each primitive comes with a type or schema that defines how it works. This can help you determine the contribution each layer is making to the stage.



Draw Mode - Here you can change the display of any element in the path to either Full Geometry, Bounding Box or Textured Cards.

Display Options- Here you can set the visibility or activation of each layer. It is not possible to delete anything from the scene graph therefore hiding or deactivating is necessary.

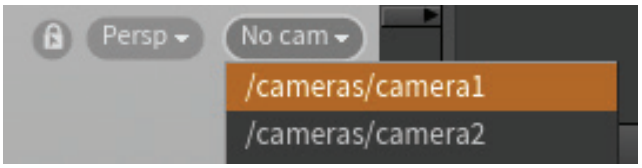
Variants- If a Layer has variants then the chosen one is displayed here.

Solaris: Cameras and Lights

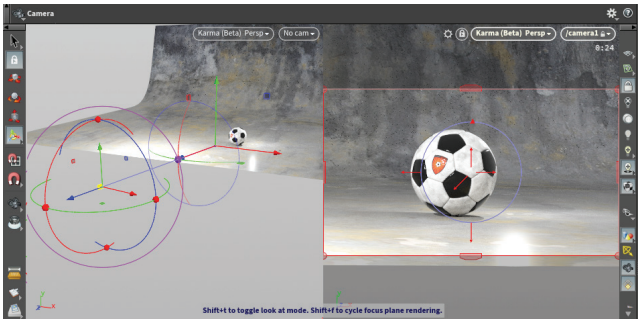
Before you render a shot, you need to look through a camera and light your scene. There are lights and cameras in the Solaris/LOPS context which is especially designed for layout, lookdev, lighting and rendering with Karma or at the object level for rendering with Mantra.

CAMERA

Cameras can be accessed from the **LOP Lights and Cameras** shelf. Press **Alt-click** on the shelf tool to convert your current view into a camera view. If you create the camera node in the Network view, click on the **No Cam** menu in the top left of the viewport to look through it.



To adjust the camera, you can use the Camera handles either from another view or while looking through the camera. There is a **lock camera to view** button in the Display Options bar that lets you use the **View tools** to reposition the camera.



There is a **lock camera to view** button in the Display Options bar that lets you use the **View tools** to reposition the camera. Just make sure that you don't leave this on later otherwise your camera view may get changed by a simple view change. Once you like a camera view, you may want to **lock its transform values** to avoid losing it to a view change.

CAMERA PROPERTIES

The Cameras in the LOP Context have key properties that determine how the camera will generate images.

View Tab

- **Projection** - Choose between perspective or orthogonal projection.
- **Focal Length** - Determines the length of the lens. Smaller values create wider shots and higher values create long shots.
- **Horizontal/Vertical Aperture** - Aperture is a gate that controls how much light goes into your camera.

Sampling Tab

- **Shutter Open/Close** - Determines how long the shutter is open which has an effect on motion blur.
- **Focus Distance** - Distance from the camera to the focal plane. Determines which objects are in focus when using Depth of Field.
- **F-Stop** - Lens aperture. Default is 0, which turns off focusing.

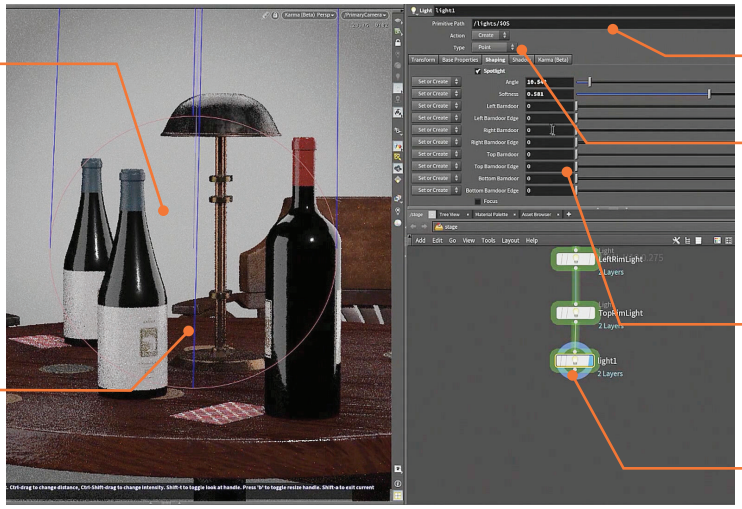
Press **Shift + F** to show a visualization of the focal plane and how it intersects geometry. When looking through a camera, you can **Shift-click** or drag on a surface to move the focal plane to intersect that point. Otherwise, you can use a handle on the focal plane to move it and set your Depth of Field.

LIGHTING SETUP

Viewport Rendering - To make lighting decisions, it is important be able to render in the viewport using either Karma or other renderers such as RenderMan or Arnold.

You can also use HoudiniGL but this is not as effective as one of the main renderers.

Light Handles - You can step back and manipulate lights in the view in a similar manner to cameras or you can use special controls to set lights right in the camera view.



Primitive Path - This sets where your light will be in the USD scene graph.

Light Type - You can choose your light type from this menu. The Light LOP gives you access to all the light types and you can switch between them.

Light Parameters - There are a range of parameters for controlling light properties such as cone angle and intensity.

Light LOP Nodes - Every light is added into the network as a LOP node.

LIGHTS

Lights can also be accessed from the shelf and have similar handles to help you position them. There are a number of different light types in Houdini to choose from.

Point Light - Emits light from a point in all directions and is similar to a light bulb.

Spot Light - Radiates a cone shaped beam of light from a point in a certain direction.

Area Light- Automatically distributes a number of light sources over a specified area. There are five area light shapes to choose from: Line, Tube, Grid, Disk an Sphere.

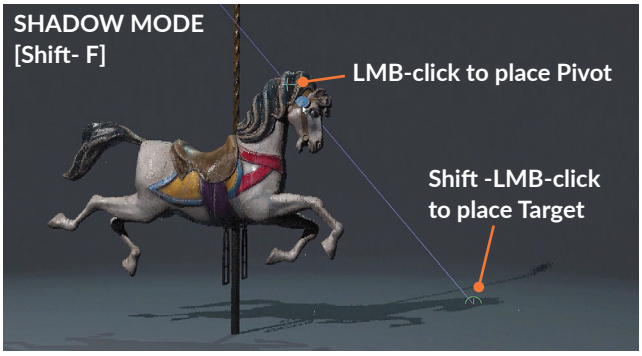
Geometry Light- This object emits light into the scene using a geometry object's surface shader for the colors of the emitted light.

Distant Light- Emits parallel rays of light, which are similar to the rays of the sun.

Environment Light- Casts light into the scene as if from a surrounding hemisphere or sphere.

LIGHTING WHILE IN THE CAMERA VIEW

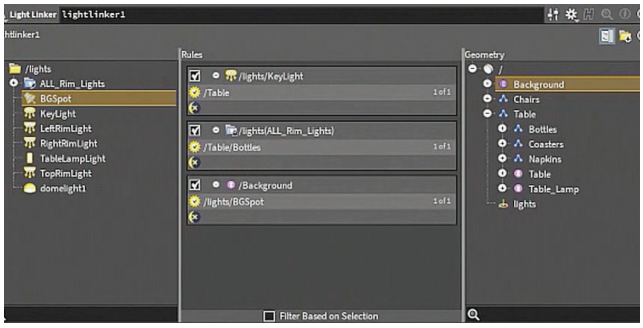
When you have a **Light** or **Light Mixer LOP** selected and displayed, you can set many of its properties while looking through the camera. You can use the **Specular [Shift-S]**, **Diffuse [Shift-D]** or **Shadow [Shift-F]** options that let you click on surfaces in the scene to set up the light.



You can then use **Ctrl-drag** to change the **distance** of the light from your shot and **Ctrl-Shift-drag** to change the **brightness**. Doing this in the viewport lets you stay focused on the shot you are working on instead of pulling away to drag on handles.

LIGHT LINKER

Linking lights to specific objects is a great way to control the lighting of the shot and this can be accomplished in Solaris using a **Light Linker LOP**. This node includes an interface for making the object/light connections that you need.



You can use collections of lights to apply the linking more efficiently using rules that define the interactions between primitives and lights.

INSTANCING LIGHTS

Within the Solaris/LOP context, you can use Houdini's procedural geometry nodes to create points then **Instance Lights** to those points. You can then add attributes to the points to create effects such as a rotating intensity you might find in an old school marquee. This approach makes it easier to set up the lights and even easier to add effects and make changes to meet the needs of the shot.

LIGHT MIXER

Light List - This list shows all the lights feeding into the mixer.

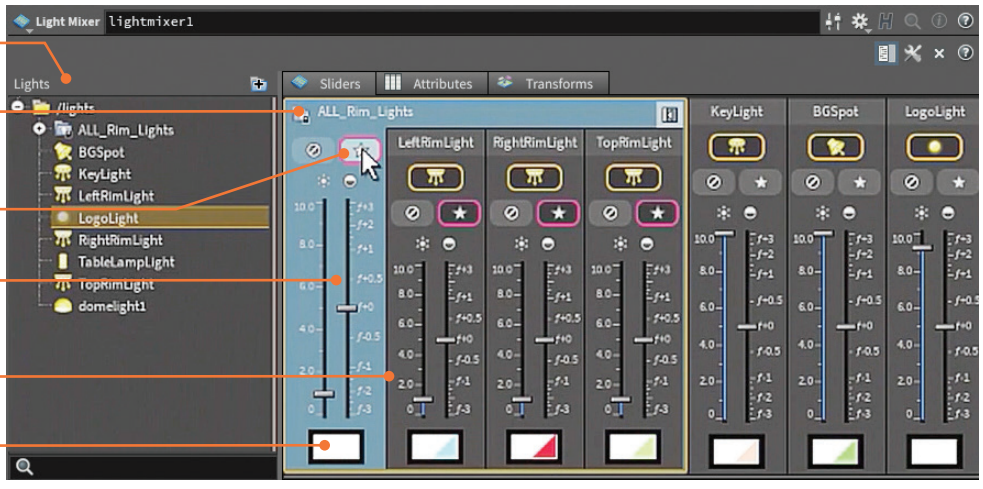
Collection - You can organize lights into collections so that they work as a group in the mixer.

Solo Lights - Click on the star icon to solor the light or the collection.

Intensity Slider - The first slider gives you control over the intensity of your lights or light collections.

Exposure Slider - The second slider gives you control over exposure which offers a tweak to the intensity.

Light Color - Click here to tint the light or the collection of lights.



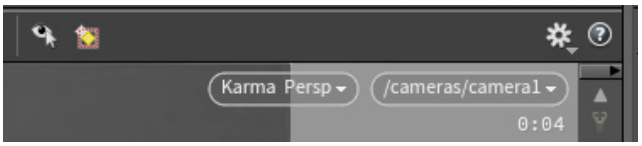
Rendering

When you render a shot, you are digitally photographing or filming your 3D objects using cameras and lights to generate either an image or a sequence of images. Game artists may also use rendering to render game cinematics or to bake textures from a high resolution to a low resolution object.

KARMA

Karma is a physically-based ray-tracer built to work with USD files in the Solaris/LOPS context. It works on the CPU and includes features such as adaptive tesselation for displacement and subdivision surfaces, multi-segment motion blur, instancing, hair & fur and strong volume rendering support.

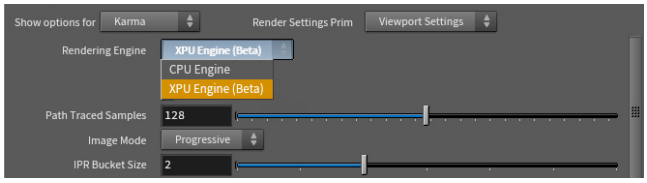
Karma works with **Hydra**,the USD imaging framework. This makes it available in the Viewport for interactive updates or you can render to disk using a Karma node.



Karma works with shaders created using VEX, USD preview and Material X.

KARMA XPU

Houdini 19.5 includes a beta version of the **Karma XPU** render engine. This hybrid GPU/CPU renderer is being released in Alpha. Many features are still under development and this engine is for testing purposes only. You can choose XPU in the Scene view's Display Options or in the Karma node.



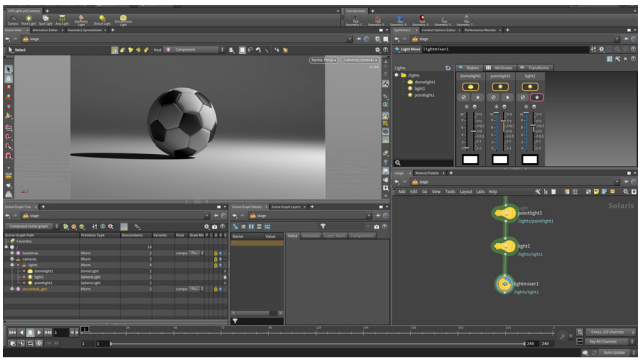
Karma XPU works with USD preview shaders and Material X but does not work with VEX.

3RD PARTY RENDERERS

With its support for USD, Solaris makes it possible to render to other Hydra delegates such as **RenderMan**, **Autodesk Arnold**, **V-Ray**, **Maxon RedShift** and **AMD ProRender**.

VIEWPORT RENDERING

One of the key benefits of the Karma renderer is the ability to use it in the perspective view. By choosing Karma, you get interactive rendering that helps you make lighting and lookdev decisions within the Solaris/LOPS context.

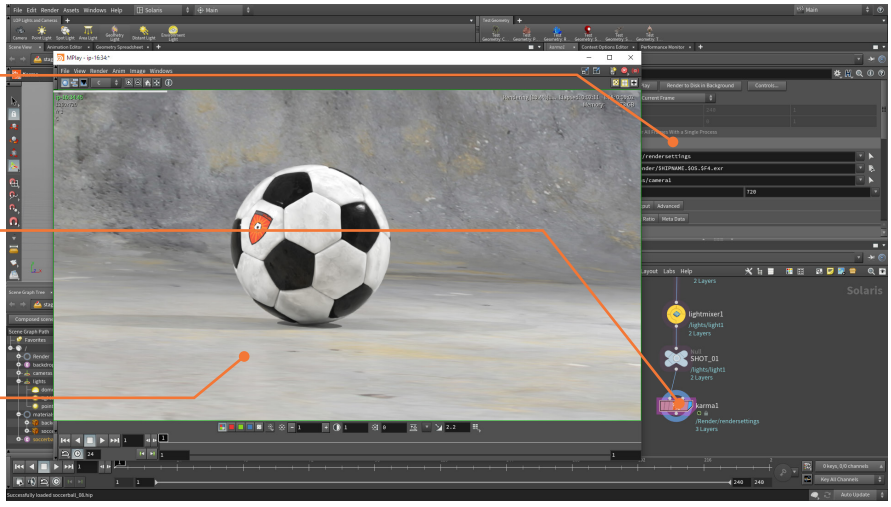


VIEWPORT | KARMA

Render Settings - To render to disk, you can use the Karma LOP to define your render settings. Here you can set a path for rendering to disk, camera settings and more.

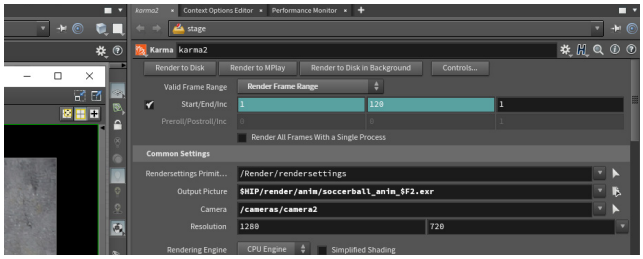
Karma LOP - Add the Karma LOP to the end of a Solaris node network. Different versions of this LOP can be used to set up unique results such as test renderings or final shots.

MPlay - You can either render from both Karma and Mantra to MPlay directly or render to disk then open up with MPlay to review the results.



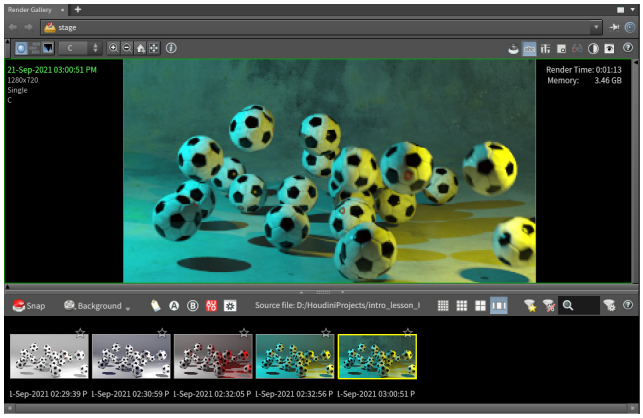
RENDER SETTINGS

When rendering the Stage, you use the viewport render settings. To create the final look of your rendering, the **Karma LOP** can be used to set a frame range, camera resolution, denoiser and higher quality render settings.



RENDER GALLERY

The **Render Gallery** lets you take **Snapshots** of your progress. Each snapshot contains all the settings of the look and at any time the scene can be reverted to match the snapshot. Snapshots can then be labelled and filtered for easy access.

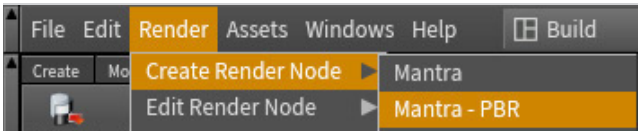


MANTRA

Mantra is the Houdini renderer developed before Solaris was introduced. It is a physically-based rendering engine that is deeply integrated with highly efficient rendering of geometry, instances and volumes but it does not work in Solaris/LOPS.

OUTPUT NODES

To render out a shot, you will need to create a render output node. You can do this by choosing **Render > Create Render Node > Mantra - PBR**.



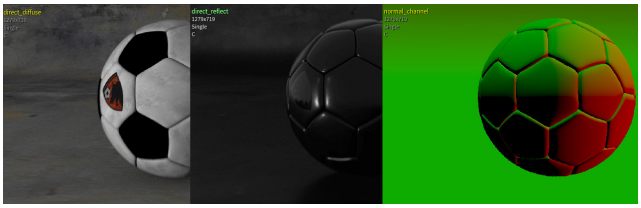
You can also add a **Karma ROP** to the output network using the **tab** key. This has a LOP network inside it that grabs all the visible objects from the object level.

You can use ROP nodes to render to disk or to Mplay. These nodes contain many of the parameters that you need to control the final image such as sampling, noise level and overall render quality.

It is possible to have a different ROP per object or group of objects and you can use them to select Mattes and Phantom objects. You can create ROP dependencies by wiring different nodes together. If you press the render button on the last node in the chain then all the other nodes will render first.

RENDER OUTPUTS | AOVS

On the ROP, you will find controls for setting up **Image Planes** to create render layers for *Direct lighting*, *Indirect lighting*, *Shadows*, *Depth* etc. Both Karma and Mantra can render out these passes which can be loaded in Houdini's compositing context, COPS, or in an external compositor such as Nuke.



The **Background Plate LOP** can be used to matte objects in order to leave holes in the scene to make the background visible. This geometry can take shadows and contribute to reflections to create a more realistic fit for your objects.

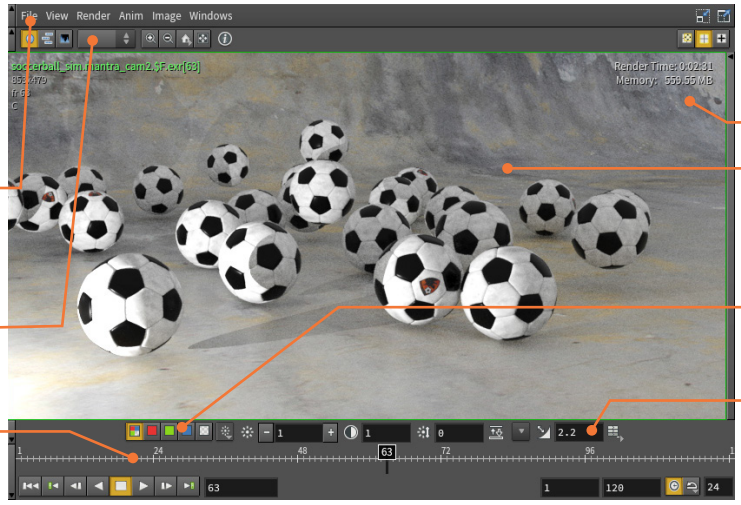
MPLAY

MPlay lets you view images rendered in Karma, Mantra or other renderers.

Main Menu - This menu lets you load images or sequences of images to preview them. You can also save them out to another format if needed.

Render Layers - This menu lets you display different render layers such as *color*, *normal*, *diffuse_direct* or *reflect-direct*.

Playbar - If you have loaded up a sequence of images then you can use these controls to play and scrub through the sequence.



Render Time - Since you are sometimes rendering directly to this view, render time info will be displayed.

View Options - **MMB-drag** to pan and **RMB-drag** to zoom in and out of the image.

Channels - You can click on these buttons to focus on red, green, blue or alpha channels or to see them combined.

Gamma Settings - Set the brightness, contrast and gamma for the viewport. By default a gamma of 2.2 is used to support a linear workflow.

Time & Motion

Animation involves changes happening over time. Whether it is an object's position, shape or color, once changes occur over time, you are animating. Houdini has a variety of tools for keyframe-based workflows in addition to **Motion FX & CHOPs** for more advanced manipulation of time and motion.

SETTING KEYFRAMES

Keyframes let you set specific parameter values at a specific points in time. As these values change, the objects in your scene are animated. You can then use animation curves to determine the in-between quality of the motion. Here are a few main hotkeys used to set keyframes on your objects while working in the scene view:

- Set Keyframe — K
- Toggle AutoKey — Alt + K
- Key Handle — Ctrl + K
- Key Translate — Shift + T
- Key Rotate — Shift +R
- Key Scale — Shift + E

You can also set keyframes in the parameter pane by pressing the **Alt** key and clicking on either a parameter name or parameter field or by **RMB-clicking** on a parameter and selecting **Keyframes > Set Keyframe**. This lets you set keyframes one parameter at a time

THE PLAYBAR

The **Playbar** is at the bottom of the main workspace and lets you playback and scrub through your animations. Time is measured in frames with a default rate of 24 frames per second.

At the left are the playback controls. Here are some hotkeys for quickly setting up playback and moving through time.

- Play Forward — ↑
- Play Back — ↓
- Next Frame — →
- Previous Frame — ←
- First Frame — Ctrl + ↑
- Next Keyframe — Ctrl + →
- Previous Keyframe — Ctrl + ←

The Playbar can also be used to edit keyframes. You can **RMB-click** on the frame range in the Playbar to access options such as **Cut**, **Copy** and **Paste** of keys along with special pasting such as **Replace**, **Cycle**, **Repeat** and **Stretch**. All these options also have their own hotkeys which you can find on the menu. You can accomplish a lot working in the Playbar before moving to the **Animation Editor**.

CHANNELS

When you set a keyframe or display animation curves in the Animation Editor, you are working with **channels**.

If you select an object that has keyframed channels then they become active and the keyframes are loaded into the **Playbar** or **Animation Editor**. If you deselect the object then the channels will no longer be selected unless you pin them.

You can pin channels using the **Channel List** which is available on the right side of the **Playbar**, the left side of the **Animation Editor** or in the **Channel List** pane. In this list, you can select one or more channels to refine which of them you are keyframing and editing.

CHANNEL LIST PANE

The **Channel List** pane, lets you work with **Channel Groups, Animation Layers** and **Active Channels**. You can use the list to create groups of channels so that they can be more easily accessed. You can also use the groups to pin channels so that they are available even if you select different objects. This is a useful pane if you are setting keyframes on characters.

FLIPBOOK

As you animate your scene, you will want to preview the results in motion. The **Flipbook** tool found on the toolbar on the left side of the Scene view lets you capture frames from the viewport then playback the results as a movie in realtime.

PLAYBAR

The Playbar is where you scrub through time and create and edit keyframes. The Playbar can also be used to make quick edits, while the Animation Editor is used for more comprehensive refinements.

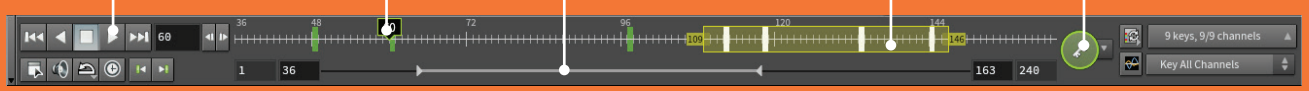
Playback Controls
These controls let you quickly play, pause, and move to next key. Below that are buttons for Animation options and Real Time playback.

Current Time
The current time is shown in the field and on the black marker in the frame range. The marker can be used to scrub through the Playbar.

Frame Range
The overall range is defined by the Animation controls button at the far left. Then the visible range can be reduced using the handles at the bottom of the range.

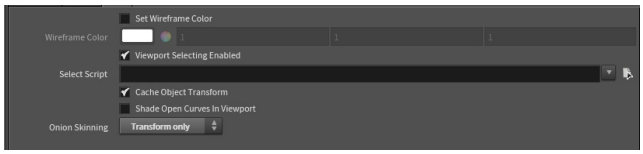
Edit Keys
As you set keys, they are shown in the Playbar. Press the **shift** key to select keys with the **LMB** and then edit the keys by dragging with the **MMB**.

Set Key
The **Set Key** button will set keyframes when you click here. Click on the small arrow to bring up a menu of options such as **Auto key**.



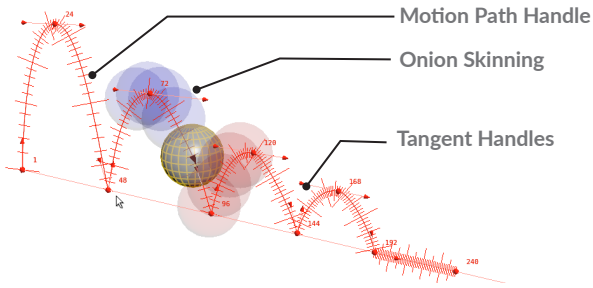
ONION SKINNING

Onion Skinning lets you display a ghosted version of your object on the frames before and after the current frame. Turn it on using the **Misc** tab on your object while the onion skinning options such as Frames Before, Frame After and Frame Increment can be found in the viewport's Display panel [**d**] under the **Scene** tab.



MOTION PATH HANDLES

When using the **Pose** tool to animate, you can click on the **Motion Path** option to bring up handles that let you see the animation of the selected object over time. You can also use the handle to manipulate the shape of the motion.



ANIMATION EDITOR

Selected channels are loaded into the animation editor where they are represented as keyframes and animation curves, or as a spreadsheet or a dope sheet. The keyframes can be selected and edited and the curves can be shaped using tangent handles. The curves define the motion in-between the keyframes and play a key role in defining the quality of the motion.

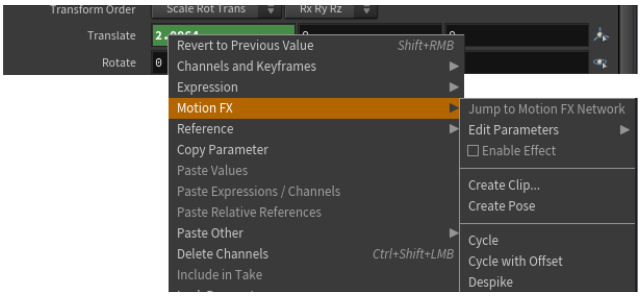
While working with channels, you can view the keyframes and animation curves using these hotkeys:

- **View All/Home** H
- **Pan** MMB
- **Zoom** RMB

MOTION FX

While keyframes and animation curves are stored in the parameters of your nodes, you can also use **channel operators (CHOPs)** for a more procedural node-based approach to working with motion.

The easiest way to create a channel operator is to **RMB-click** on a parameter and select from the **Motion FX** sub-menu. You can also apply these effects to channel groups when working in the Channel List.



Motion FX can be applied to keyframed motion which is extracted and stored in a **Channel CHOP**. You can then apply effects such as **cycle, noise, smooth, limit** or **lag** to the existing motion. On the **Constraints** shelf, you have tools which let you have one parameter either **look at, lag** or **jiggle** behind another.

This non-linear approach to working with motion offers a unique way of working that can be very flexible.

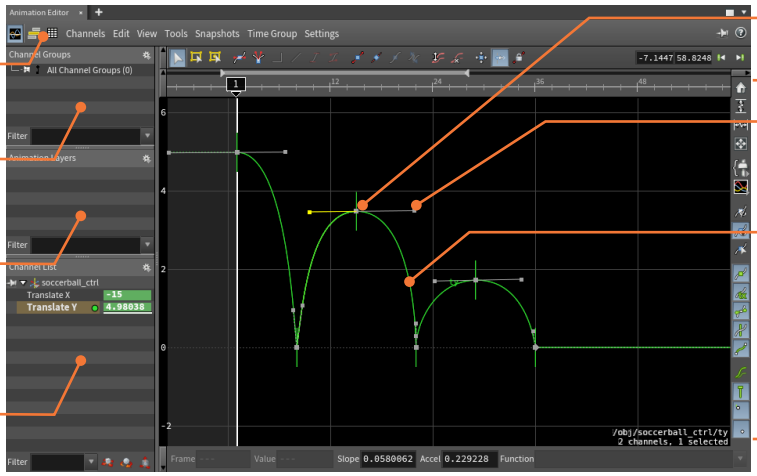
ANIMATION EDITOR

Editor Options - This editor can be shown as a graph, a dope sheet or a table.

Channel Groups - This area of the graph shows channel groups which make it easier to select and pin channels.

Animation Layers - This area lets you layer different channels on top of each other to create multiple iterations.

Channel List - Channels on selected objects show up in this area. You can then select the names of channels you want to see in the graph.



Key Handles - Move the key back and forth in time using the vertical bar or edit its value using the box.

Tangent Handles - They define the tangents coming in and out of a keyframe to help you shape the curves.

Curve - The animation curves determines the motion in-between the keyframes, which defines the quality of the motion.

Curve Functions - These let you set various display options for the animation curves and handles.

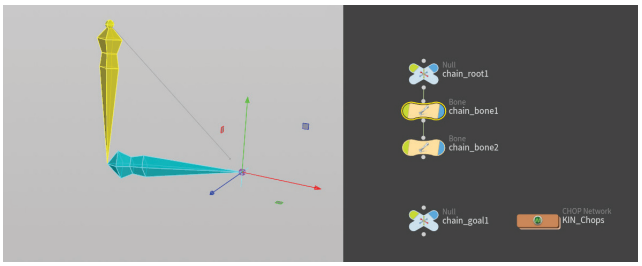
Character Rigging & FX

Houdini includes a wide range of rigging tools for creating characters and creatures that can then be wrapped into Houdini Digital Assets to be handed off to animators. Houdini also has Character FX tools such as hair, fur, muscles, cloth and crowds to enhance the look of your characters.

BONES

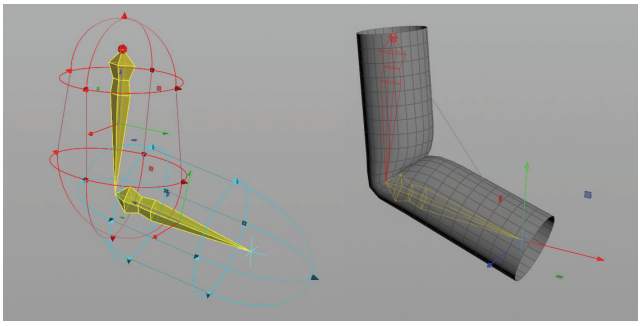
In Houdini, you draw and edit bones using the **Bones** tool and the **Bones from Curve** tool found on the **Rigging** shelf. Each bone chain is made up of a chain root and bones. Whereas other 3D apps are joint-based, Houdini uses **Bone** nodes that have parameters for **Length** and **Rest Angle**.

You can also use the **Bones** tool to add inverse kinematics to the chain which adds an **end effector** and in some cases a **twist effector**. The kinematics are driven by **Channel Operator** or **CHOP** nodes which exist in their own subnetwork.



CAPTURING GEOMETRY

The character's geometry can then be **captured** to the bones to create the deformations needed to convey realistic motion. Houdini bones include **Capture Regions** that you can set up to encompass your geometry while creating the right overlap at the joints. This process results in weighted attributes being assigned to the points that get fed into a **Deform** node that controls the geometry when the bones are moved and rotated.



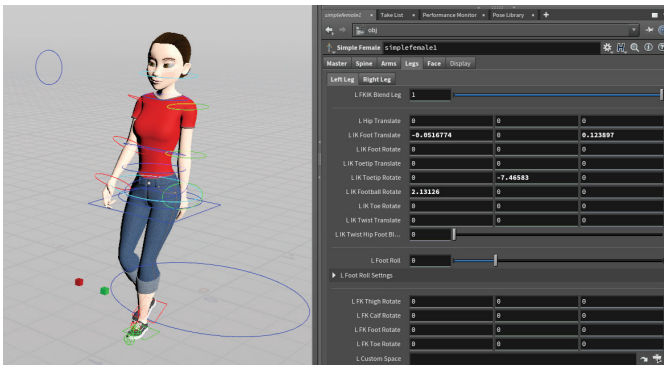
At first, captured geometry might not get you exactly the look you want, therefore you would use various tools to **Edit and Paint Capture Weights**. This lets you smooth out the weighting at the joints to create more realistic bending. You can also smooth out the effect of point deformations using a **Delta Mush** node that wires into the **Deform** node.

A new technique called **Bone Capture Biharmonic** allows you to capture geometry without requiring extensive point weights to get a desirable look at the joints. This method sets up biharmonic functions on a tetrahedral mesh to create a much more holistic solution.

DIGITAL ASSET CHARACTERS

To rig a Houdini character and share it with the animation team, you will need to wrap up the bones, geometry and materials into a **Houdini Digital Asset**.

This creates a file on disk that can be easily referenced by animators into multiple shots. Handles and key parameters are made available at the top level so that Animators can set keyframes without worrying about the inner workings of the rig. You can also save Pose Library and Character picker setups into the asset for quick access.

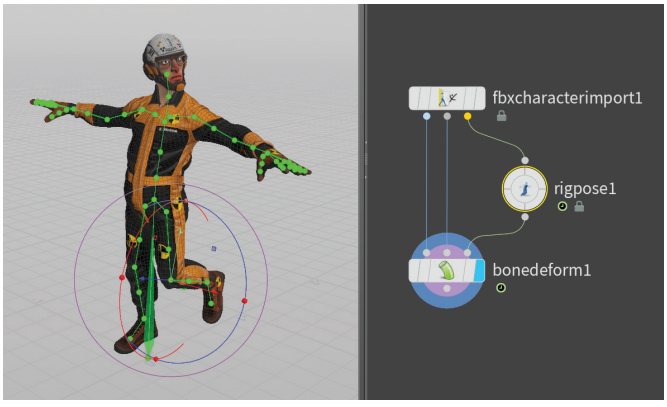


Changes made to any part of that character are saved into the asset and all shots will be updated. This creates a robust character pipeline that is easy to manage.

KINEFX

KineFX is a character toolset designed to provide a procedural foundation for retargeting, motion editing, and in future releases, rigging and animation. Set in the geometry context, these new workflows make rigging a fast, plug-and-play experience with unlimited flexibility and caching capabilities.

Implemented in the geometry [SOP] context, KineFX treats joints as regular point geometry with edge connections defining the rig hierarchy. You can bring rigs in from the object level of Houdini or import FBX characters.



CHANNEL GROUPS

When you animate in Houdini, channels that are scoped can be keyframed and displayed in the **Animation Editor.** Generally these channels are based on your current selection. You can also put together **Channel Groups** that let you scope and pin down channels to assist with keyframing.

When you have a character set up as a Digital Asset, you can click on its icon at the top left of the parameter pane and choose **Parameters and Channels > Create Nested Channel Groups** to create groups using the asset's folder hierarchy as a guide. A well designed character asset makes this easy.

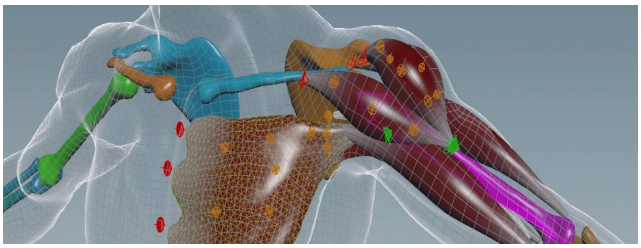
CHARACTER FX | HAIR & FUR

Houdini has a hair and fur toolset that you can use to setup and groom your character starting with the **Add Hair** tool. These tools also let you work with guide hairs and then animate them using wire sims to create added realism.

CHARACTER FX | MUSCLES & SKIN

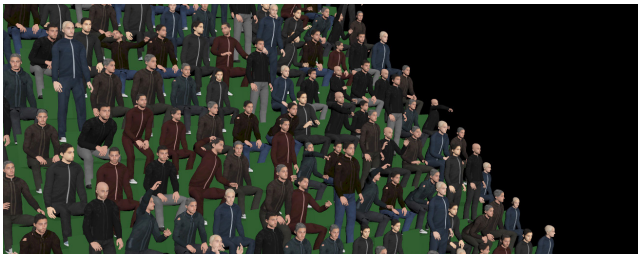
Houdini lets you add muscles to an animated creature and then apply them as a skin deformer without requiring any simulation. Start by creating simple muscle forms using the **Muscle** nodes in the geometry context.

You then adjust the muscle's shape and placement, attach it to your character rig then enable automatic secondary animation or jiggle. Houdini's muscle system has been designed to serve both FEM (dynamics simulations) and non-FEM (skin deformer) workflows while using a unified set of Digital Assets.



CROWD SIMULATIONS

A crowd simulation begins with agents that are made up of a character skeleton, skin geometry, and animation clips. These are assigned to points, where simple rules can combine to create complex behaviors and the agents can interact with other dynamic elements. For example, an agent might be struck by a passing car and become a rag-doll or a crowd might be triggered to react to an action on the field.



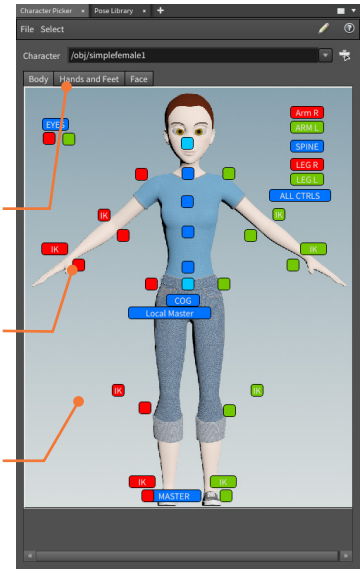
CHARACTER PICKER

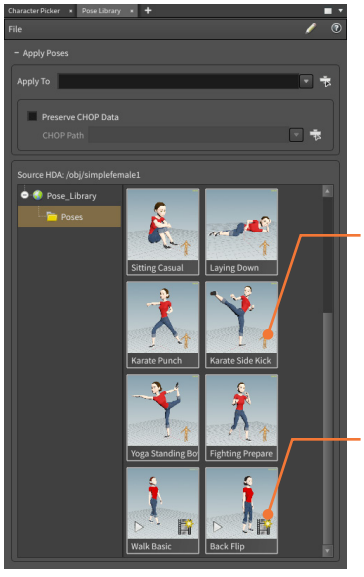
This pane lets you create an interface for choosing parts of your rig. This can then be saved into a file that you add to your Digital Asset file on disk.

Tabs - Set up multiple tabs for different areas of the body such as hands, feet or face.

Controls - Place markers for the different handles on your rig then add text or color to help you distinguish between them.

Background Image - Add a visual representation of your character to properly associate the markers with the parts of the body.





POSE LIBRARY

The Pose Library pane lets you capture poses and clips from your character for future reference. You can go to a frame in your Playbar and apply the pose by clicking on it here.

Pose - These save out a pose taken from a single frame. All of the parameter settings at that pose would be applied to the character in your current scene. You would then use this to interpolate to another pose.

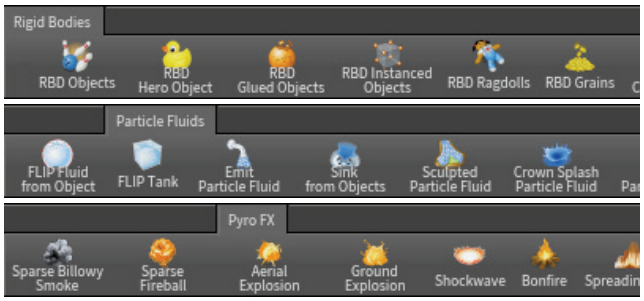
Clip - A clip contains a set of keyframes for a longer time period. These might be a walk cycle or a particular movement like a back clip.

Dynamic Simulations

Whether you are creating Bullet Rigid Body destruction, Pyro FX fire and smoke, Vellum Soft Bodies or FLIP fluids, Houdini lets you work in an integrated dynamics environment. Different solvers know how to talk to each other to allow for more directable results.

SHELF TOOLS

Setting up dynamic simulations involve a network of nodes in the **Dynamic or DOPS** context, as well as nodes at the **Geometry or SOPS** context. It is always a good idea to use the shelf tools because they will add all of these nodes for you and reduce the number of clicks needed to set up a sim. You can then dive into the networks to explore all of the nodes.



The shelf tools are great for setting up groups of nodes automatically. Seeing how these shelf-built networks are put together can be very useful later if you choose to set up DOP networks from scratch.

DYNAMIC SOLVERS

At the center of any simulation is the dynamic solver. It is the brain of the simulation and takes all of the dynamic objects, forces and collision objects and integrates them to create the final result. The shelf tools put these solvers into a Dynamic Network and wire up the nodes for you.

Rigid Body Solver- Simulate rigid objects falling and colliding using the efficient Bullet solver or Houdini's built-in solver.

Static Solver- For situations where you want objects to work as collision geometry but not be affected by the simulation.

Flip Solver - This solver creates FLIP Fluid simulations to create splashing and wave effects.

Whitewater Solver - After completing a FLIP solve, you can run this solver to create foam, spray and bubbles.

Vellum Solver- A type of POP Solver that includes integrated support for cloth, hair, grains, fluids and soft bodies such as balloons.

POP Solver- Used for particles and grains, this solver simulates a wide range of different particle-based scenarios. Grain simulations can also be used for soft body and cloth-like simulations.

Wire Solver- You can use this solver for hair and fur or other wiry objects such as the rigging of a ship or the branches of a tree.

Finite Element Solver- Simulates the physics of continuous materials or solids as determined by tetrahedrons. This solver is used for muscles, soft body sims and destructions shots such as breaking wood.

Cloth Solver- Create cloth simulations that can collide with deforming geometry such as a character.

SOP Solver- Use a SOP network to evolve an object's shape over time such as a wall being dented as it gets hit by objects.

OPENCL

You can use the GPU for faster sim times using **OpenCL** on solvers such as the **POP Grain node**, the **Pyro solver** (**Advanced** tab) and the **FLIP solver** (**Volume Motion > Solver** tab).

FORCES

To create dynamic motion, forces are needed to "get the ball rolling." The most basic of forces is gravity although other external forces such as fans, fluids and magnets can also play a role in initiating motion in your simulation.

Gravity Force A downward force on objects which works as if they were inside a gravity field.

Drag Force - Applies force and torque to oppose an object's existing motion to slow it down or dampen its momentum.

Uniform Force- A precise amount of force and torque that can be augmented by a noise DOP to add turbulence.

Fan Force- Applies a cone-shaped force on objects.

Fluid Force- Deform soft bodies such as cloth or wires with fluids.

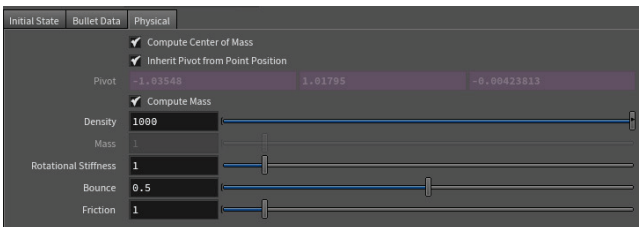
Wind Force- A pushing force that will increase the velocity of objects up to but not beyond its own speed.

Magnet Force- Attracts or repels objects using metaballs to represent a force field.

Vortex Force- Creates a vortex-like force that causes objects to orbit around a curve much like objects around a tornado.

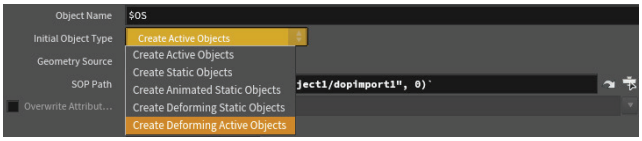
DYNAMIC OBJECTS

When you select an object and use a shelf tool to add it to your simulation, Houdini creates a Dynamic Object that uses the object's geometry and adds dynamic properties such as *density*, *friction*, and *bounciness*.



ACTIVE VS STATIC

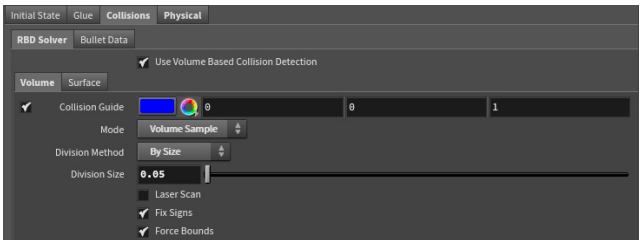
Active dynamic objects are affected by forces and collisions while Static objects are not. If you want to use animated or deforming geometry then you must define this on the dynamic object using either the **Initial Object Type** menu or the **Use Deforming Geometry** checkbox.



COLLISIONS

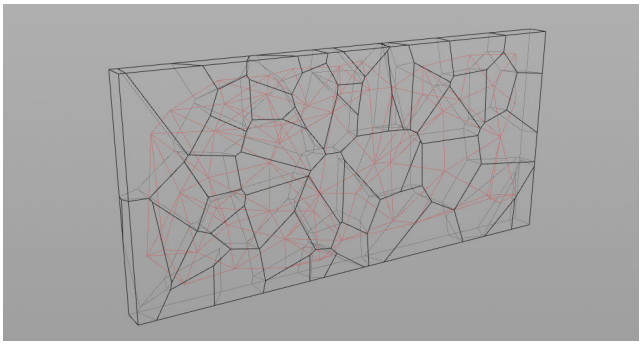
Collision objects are also a big part of any simulation. You can set up a **Ground Plane** to create a continuous surface for collisions or use either static or deforming objects.

On each **Dynamic Object**, there are also settings for displaying and optimizing the collision volume. While you often want collisions to be as accurate as possible, you need to balance that with the need for faster simulation times.



RIGID BODY CONSTRAINTS

On the Rigid Bodies shelf, you will find a number of constraints that can also be used to influence a simulation. These include **Pin**, **Spring** and **Slider** Constraints. You can also use **Glue Objects** when you set up a rigid body sim to hold objects together until you either "loosen up" the glue or a collision occurs.

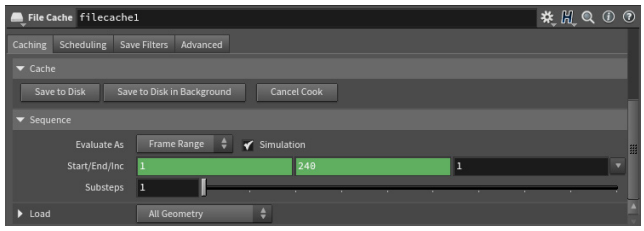


PLAYBAR FEEDBACK

To start a simulation, you press **Play** in the Playbar. As the simulation progresses, the Playbar is highlighted to show how much of the sim has been cached to memory. You can then scrub through that area without re-simming.

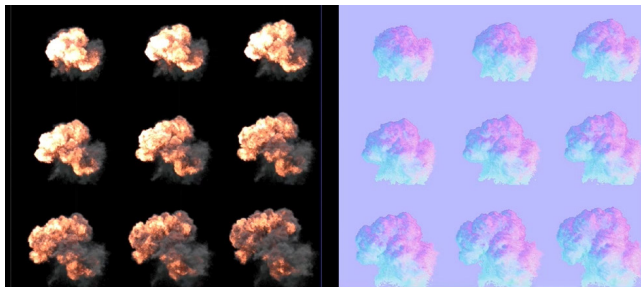
CACHING TO DISK

Once you have completed a simulation, you can either lock it down by saving out a **sim** file from within DOPs or more commonly, write out the simulated geometry to a **bgeo** sequence using the **File Cache** node. This will make it easier to work with the results of a sim during the lighting and rendering stages of production.



REALTIME FX FOR GAMES

In games, you need effects, such as explosions, to be optimized for real time in the game engine. Check out the **SideFX Labs Tools** to learn more about converting different kinds of Houdini sims such as rigid bodies, Pyro FX and Fluids into game ready art.



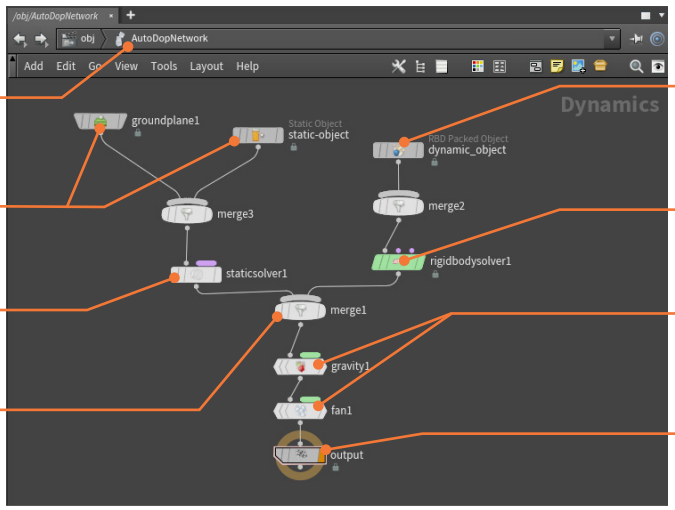
AUTODOPNETWORK

When you use a shelf tool to create a dynamic object, collision object or force, the AutoDopNetwork is created to combine all of the parts.

Static Objects - These nodes set up the properties of the ground plane and a static collision object.

Static Solver - This solver keeps the incoming objects still while dynamic objects interact with them.

Merge Node - Brings together parts of a dynamic system. During simulation nodes are evaluated up and down the chain so that everything interacts.



Dynamic Object - This node brings geometry into the DOPs environment and assigns basic properties.

Rigid Body Solver - The solver that generates the simulation of the participating objects.

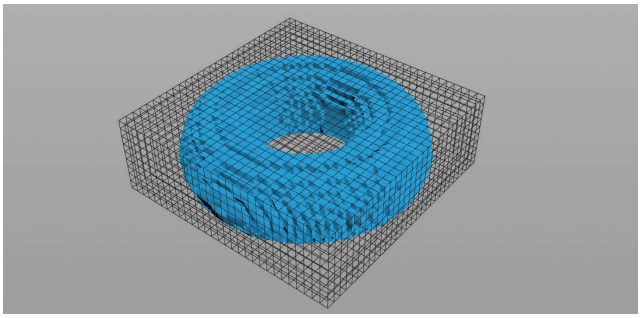
Forces - The nodes that influence the dynamic objects using forces such as gravity or wind.

Output node - You can use this node to output .sim files if you want to cache out the simulation.

Cloud FX & Volumes

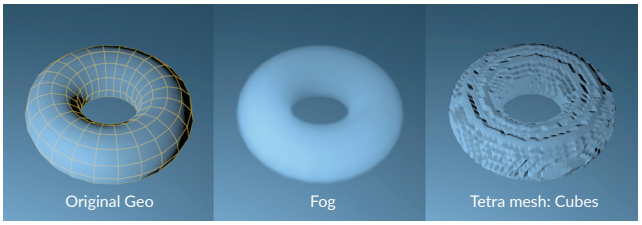
A big part of visual effects in Houdini is the use of volumetric data. In Houdini, volumes often sit under the hood to help tools get the job done but it is a good idea to learn what they are and, in time, learn how to work with them directly.

With volumes, you describe objects using *voxels* rather than points and polygons. A voxel is a 3D pixel, a cubic grid where each voxel contains information that informs how the volume will be displayed which makes it ideal for wispy cloud-like shapes. The visual quality of a volume-based object is defined by the resolution of that 3D grid. With more resolution, the results are of a higher quality but performance may be affected.



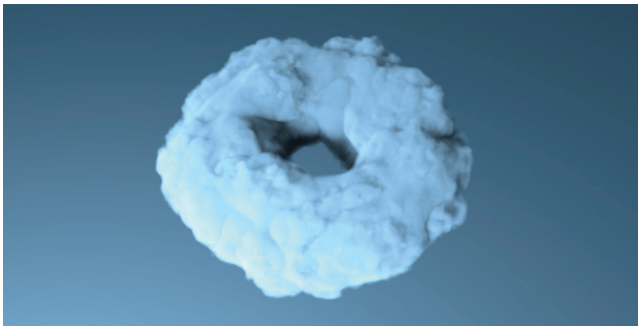
ISO OFFSET

The **Isooffset** node found in the geometry context lets you take any manifold Polygon geometry and construct a Houdini Volume for Houdini to use. You can choose from a variety of different Output types to see the shape as *fog* or a *tetra mesh*.

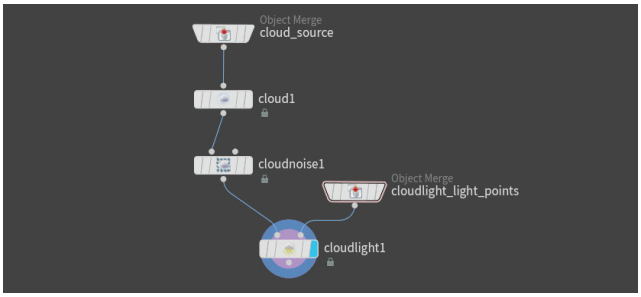


CLOUD FX

This toolset converts geometry into a cloud-like VDB volume complete with lighting. The **Cloud Rig** tool can be useful for shaping an individual cloud, or simply as a way to better understand lower-level tools such as **Cloud**, **Cloud Noise** and **Cloud light** that all contribute to the final look.



The resulting network imports the cloud source then applies these other nodes to create the cloud-like effect using Houdini Volumes and VDBs. Houdini also comes with a **Sky Rig** tool that fills the sky with volumetric clouds.



To create a cloudscape for a game engine such as **Unreal** you can use a Sky rig in Houdini, then convert it to a mesh and use it as a spawn surface. There is a tutorial on the SideFX website by **Andreas Glad** that teaches this approach.

OPEN VDB

"OpenVDB is an Academy Award-winning open-source C++ library comprising a suite of tools for the efficient storage and manipulation of sparse volumetric data discretized on three-dimensional grids. It is developed and maintained by DreamWorks Animation for use in volumetric applications typically encountered in feature film production." - **openvdb.org**

Houdini has a variety of OpenVDB Volume nodes available in geometry [SOP] networks that convert geometry into volumes.

VOLUMES UNDER THE HOOD

Many of the tools in Houdini make use of volumes under the hood where you can't see them. Here are some of the areas where volumes are making contributions to your work.

- **Colliders** - By default, volumes convert geometry into colliders for dynamic simulations.
- **Simulation Fields** - Volumes define fields such as density or velocity that contribute to dynamic simulations.
- **Hair and Fur tools** - These tools use volume data to assist with grooming calculations.
- **Terrain** - Heightfield tools use 2D volumes where each voxel contains the height of the terrain at each grid point.
- **Rendering** - Volumes create water depth and fog effects in Mantra.

Terrain & Heightfields

Procedural terrain generation in Houdini is achieved using a collection of heightfield nodes that let you layer shapes, add noise and run erosion simulations to define the look for your digital landscapes. This a workflow that is similar to compositing, but you do all your work with 3D shapes.

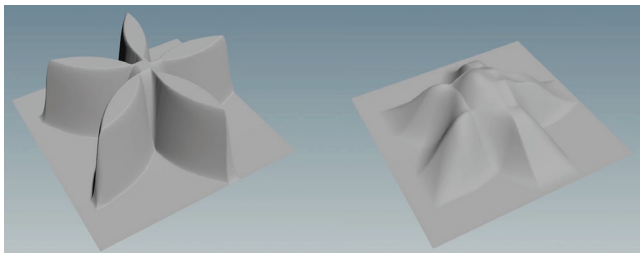
Houdini provides a variety of geometry nodes for generating and shaping terrain. These tools represent terrain using 2D volumes where each voxel contains the height of the terrain at that grid point, called **heightfields**. The data passing through a geometry network can contain multiple heightfields. You can access these tools using the **Terrain** desktop.

The Houdini viewport lets you visualize the 2D heightfield as a 3D surface, and the mask field is displayed as a red tint on the 3D surface. There is a dedicated Mantra procedural for rendering heightfields and they can be used as collision surfaces for dynamic simulations.

PATTERNS

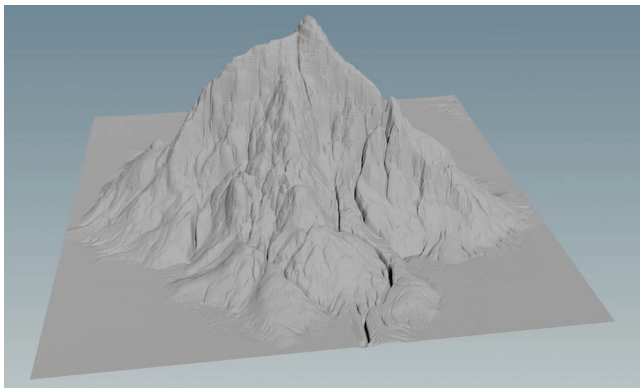
After laying down a **Heightfield** node to define your base resolution, the **Heightfield Pattern** node gives you access to a number of starter shapes. You can set up linear, concentric, or radial ramps, linear steps, radially symmetrical shapes such as stars and voronoi cells.

These shapes can then be blurred and distorted to get shapes that you can use to begin your terrain. You can also combine and layer elements to achieve even more sophisticated results.



NOISE

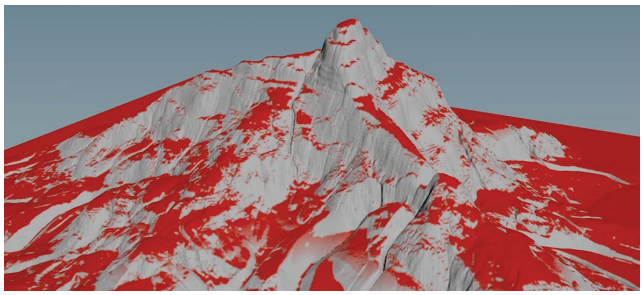
As you build up your terrain, you can then add noise to layer in a natural look. You can choose from a variety of different types of noise including, *Perlin*, *Sinusoid*, *Worley* and more. This adds realism to your terrain and by combining different shapes with different kinds of noise, you can achieve a wide variety of hyper realistic results.



MASKS

The heightfield tools also use a secondary type of 2D volume, where each voxel contains a **mask layer**. Most terrain nodes take a second input that can contain a mask layer to control which parts of the terrain the node will modify.

You can use a variety of different methods to create masks and then use them to assist you as you add detail and shape the terrain. You can also draw or paint masks onto the heightfield.



EROSION

The **Heightfield Erode** node uses rainfall, the erodibility of the soil, and entrainment rates as variables to simulate erosion and deposit buildup. This node works iteratively during playback. It will appear to have no effect on the first frame. You need to press play in the Playbar to watch as it sims the erosion.

EXPORT OPTIONS

There are a couple of different ways to export your terrain for use in another application such as a game engine. You can use the **Heightfield Output** node to export height and/or mask layers to disk as an image then bring these in as textures.

You can also create **Houdini Digital Assets** that will open up in applications such as Unreal and Unity using the Houdini Engine plug-ins. In the game engine, these Digital Assets will interact with built-in Terrain tools.

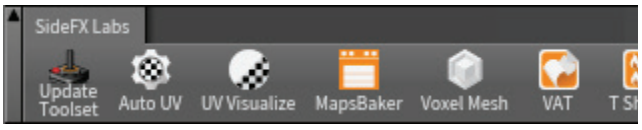
SideFX Labs

SideFX Labs is a collection of high level tools aimed to speed up artist and gamedev related workflows in Houdini. There are a growing number of tools being developed that range from Mesh Processing, Realtime FX, UVing, to creating Motion Vectors from simulations.

While all of Houdini can be used to generate content for film, TV and games, the Labs toolset addresses artist-specific tasks that may not be available in Houdini today. These tools are developed separately from the regular Houdini development cycle and become available the moment they are ready for testing. They can be downloaded directly from within Houdini or accessed through the SideFX Labs **github** page.

DOWNLOADING THE TOOLS

The Labs tools can be installed with Houdini or accessed from the SideFX Labs shelf tab. It not visible on most desktops so you will need to add it. Once it is visible, click on the **Update Toolset** button. This will pop up a dialog prompting you to install. Since many tools are in beta, you may want to turn **Off** the **Production Builds Only** option.



The shelf will be populated with many of the tools but some are only available when you **press tab** in the Viewport or Network view. These nodes are prefaced with **Labs** to make them easy to find.

FX TOOLS

Houdini is known for its strong FX tools and SideFX Labs has tools to process the results for use in a realtime environment for games or virtual production. Labs has a range of tools to help optimize and export your sims to textures, fbx, csv, etc.

Vertex Animation Textures - The Vertex Animation Textures ROP will export a mesh and textures to be used with a realtime material that can playback complex animations for cloth, rigid body destruction, fluids and particles.

Flipbooks Texture - Fast GL or Karma based creation and preview of texture atlases for Pyro FX.

Destruction Cleanup - Prepare rigid body simulation results for export, reducing redundant geometry, cleaning normals, cleaning attributes.

Skinning Converter- Skinning Converter is a SOP that can convert any non-changing topology deforming mesh sequence into a bone based animation.

Make Loop - Takes a mesh, points or volume that is animated and loops them.

Volume to Texture - The Volume Texture tool lets you write out a texture that can be used with Ryan Brucks' volume plugin in UE4.

Flowmap - This utility tool sets up a flowmap template on your input geometry.

Flowmap Visualize - A high quality realtime preview of a flowmap texture in the Houdini viewport.

Flowmap Obstacle - The flowmap obstacle SOP allows for easy modifications on the flowmap based on geometry.

Niagara ROP - All-in-one HDA to extract and write out impacts, split data and interpolation data from a bullet sim to be used with the UE4 Niagara data interface.

Gamedev Procedural Smoke - The procedural smoke SOP will generate an animated volume to represent smoke.

ROP Vector Field - Generate UE4 compatible vector fields from volumes or point clouds.

MESH PROCESSING

There are a lot of steps to get quality meshes into your games - photogrammetry, topology cleanup, mesh reduction, uv layout, baking maps. Lab tools let you simplify that process with wrapped up workflows and integrations with popular software.

AliceVision Photogrammetry - AliceVision is a Photogrammetric Computer Vision Framework which provides a 3D Reconstruction and Camera Tracking algorithms.

ZBrush Bridge - GoZ is Zbrush's fast file transfer feature that lets you send meshes between Houdini and Zbrush seamlessly without having to deal with file paths or extensions.

Delete Small Parts - Removes parts based on connectivity and size.

Delight - Removes ambient lighting information found in high-res photogrammetry scans.

GameRes - Full Pipeline Node to Take High Res Models to Low Res.

Maps Baker - Generates textures bakes from a high resolution to low resolution model at near interactive speeds.

LOD Hierarchy - Create and export an LOD Hierarchy as FBX.

Mesh Sharpen - This tool sharpens the geometry based on the curvature found on the mesh.

Edge Damage - This tool will add edge wear to your geometry.

WORLD BUILDING

Digital worlds are becoming bigger and more complex and it is important to have an efficient world building workflow. Whether you want to recreate New York City, grow a dense forest or add interior details to your sci-fi adventure there are Lab tools for you.

Physics Painter - Physics Painter is a SOP that allows users to paint physics objects onto any other object.

Building Generator - Convert low-resolution blockout geometry into detailed buildings using a library of user defined modules.

OSM Import - Open Street Map is a great database for city street data. This node efficiently loads the OSM files into Houdini as well as all of the different tagged attributes on the buildings and streets.

OSM Buildings - Generate Buildings from OSM Data.

Tree Tools - The Tree Tools in Labs consist out of several tools that can be used together to create intricate branching structures such as trees, bushes, plants and much more.

Cable Generator - Given a curve that represents the high 'pin' points and low 'sag' points of a cable, this sop will generate sagging cables, with user definable cable count, shape, color.

Curve Branches- Scatters curves over curves, with many intuitive controls to go from clean geometric branches to organic vines. Duplicates of this sop can be chained together for recursive growth, approximating the look of L-Systems but much more controllable.

Dirt Skirt - Create a geometry 'skirt' where an object and ground plane intersect, to be used as a soft blend in a game engine.

Lot Subdivision - Divide polygons into panels. Useful for city blocks or greeble.

MapBox - Generate color, height and Open Street Map (OSM) curves using data provided by mapbox.com.

SciFi Panels - Example HDA to generate Sci Fi Paneling.

Snow Buildup - Adds geometry to an input mesh to mimic the build up of snow.

Terrain Texture Output - The Terrain Texture Rop SOP renders image data from a heightfield.

MODELING

The Labs Tools includes a variety of modeling tools designed to make it easier to create game-ready geometry.

Decal Projector - Project a decal (localized piece of geometry and a texture) onto geometry.

Calculate Slope - Calculate the slope of a surface by comparing to a direction, and optionally blur and remap the result.

Curve Sweep - Sweep a profile along input curves, with simple controls for profile type, width, and twist behavior.

Extract Silhouette - Create an outline of an object projected from one of the axes, xyz.

UV MAPPING

Texture UVs are a big part of creating game art and these tools augment Houdini's existing UV toolset to make you faster and more efficient.

Auto UV - Automatically generates the seams for an object and immediately runs a UV Flatten after the fact.

Inside Face UVs - Create UV's for inside faces of fractured geometry.

UV Transfer - Transfer uv's between a source and target geometry.

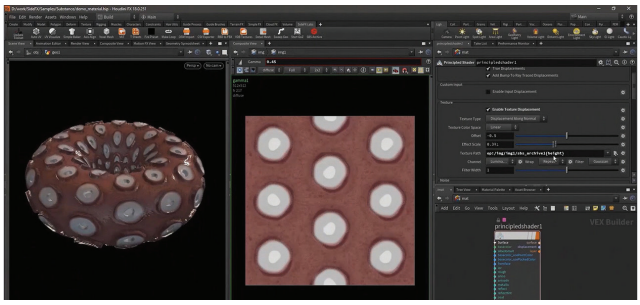
UV Visualize - Helper script to visualize UVs. Including features such as: Visualize Seams, Warp between UV space and Model Space, Modify the tiling of the grid texture and Visualize Islands.

Texel Density - This tool calculates the current texel density of an asset per primitive based on the asset and project resolution.

INTEGRATIONS

These tools make it easier to import and export data into Houdini and out to game engines.

Substance COP - The Substance Plugin for Houdini lets you load Substance Archive files into Houdini in COPs.



Rizom UV- The RizomUV Bridge is a set of 4 different SOPs that facilitate the communication between Houdini and RizomUV.

Quad Remesher - The QuadRemesher node is a wrapper around Exoside's QuadRemesher command line interface.

Instant Meshes - Reads in DDS (DirectDraw Surface) Files.

Sketchfab - Uploads geometry to Sketchfab.

3D Facebook Image - Quickly render a 3D scene to a 2.5D image that can be uploaded to Facebook.

Marmoset ROP- The Marmoset ROP allows you to quickly generate an mview inside Houdini.

Gaea Tor Processor - The Gaea Tor Processor allows you to load up build .TOR files made in Gaea.

UX

Some Labs tools are specifically designed to enhance the user experience for artists working with Houdini.

Crash Recovery - Found under the File> menu, this functionality allows for quick recovery from an unfortunate crashed file.

Network Paint - Allows for colorful annotations of the network by simply drawing in the network editor.

Sticker Placer - Helpful for annotating networks by placing numbers, icons and user-created graphics.

External Script Editor - Sets up a live connection with an external IDE when working with VEX, Python, OpenCL and expressions.

AND MUCH MORE...

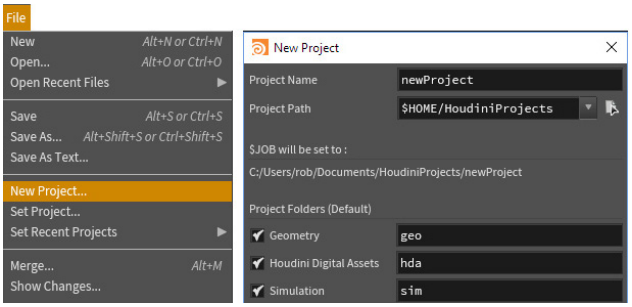
More Tools are being added on a regular basis. Go to *SideFX.com/labs*

File Management

Understanding how to manage all of the files you create while working with Houdini is very important to your success as an artist. A typical scene file can have outside dependencies on disk and managing these is important especially if you are moving your files to a different computer.

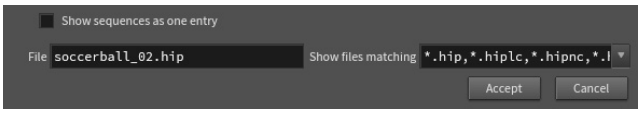
PROJECT DIRECTORIES

While Houdini can work with files scattered all over your hard drive, this will make it harder to share your work and manage file dependencies. It is better to set up project directories using **File > New Project** or use **File > Set Project** to choose an existing project directory as the home base for your work. This will make it easier to set up local dependencies with respect to all of the required project files.



SCENE FILES | .HIP

The main file type when working with Houdini is the **.hip** file. This file contains all your nodes and networks and is the file type used when you save your work.

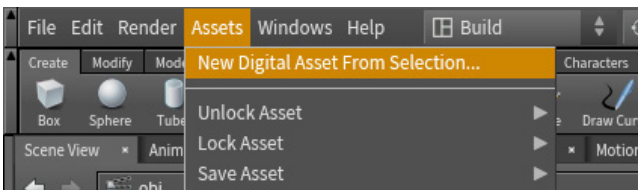


UNIVERSAL SCENE DESCRIPTION | USD

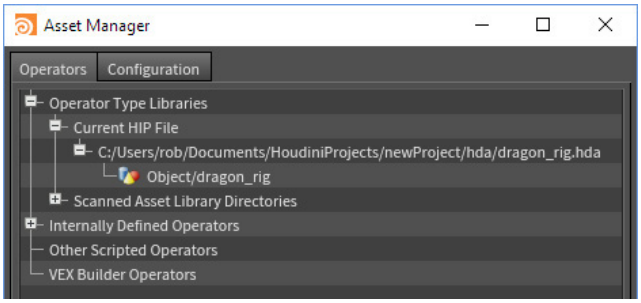
In Houdini, the Solaris lighting and lookdev environment works with USD [**Universal Scene Description**] which is an open source initiative created by PIXAR. In Solaris, USD is native and you can use procedural nodes to manage references, payloads, layers, collections, variants and level of detail.

HOUDINI DIGITAL ASSETS | .HDA

You can also encapsulate then save out Houdini networks into **Houdini Digital Asset** or **.hda** files. Parameters from inside the asset can then be promoted to the top level to create a custom UI for the asset. These files can be easily shared with other artists and provide a robust referencing architecture as your assets evolve through the life cycle of a project.

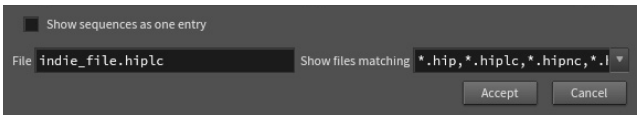


To create and load assets you can use the **Asset** menu. You can also manage assets loaded into your scene using the **Asset Manager** found on that menu. If you have two HDA files loaded into your scene that have the same name, Houdini will choose one of them based on rules set up in the manager. Changes made to an asset definition in an HDA file will be automatically pulled into scenes that reference that file. Note that older Digital Asset files may have a .otl extension which will work exactly the same as .hda files.



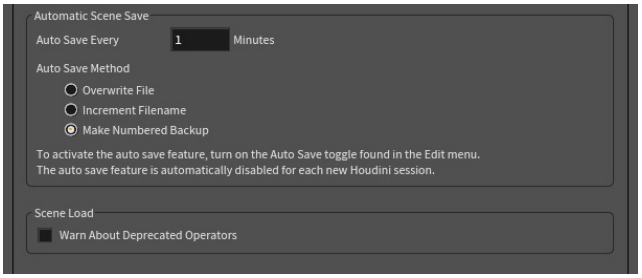
APPRENTICE AND INDIE FILES

Houdini Apprentice and **Houdini Indie** use different file types that cannot be opened in a commercial version of Houdini. Apprentice uses **.hipnc** (non-commercial) for scenes and **.hdanc** files for assets and Indie uses **.hiplc** (limited commercial) for scenes and **.hdalc** files for assets.



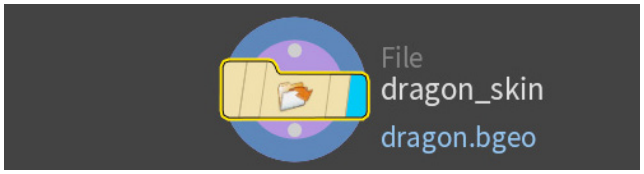
BACKING UP YOUR WORK

By default, Houdini creates a numbered backup of your scene files and Digital Asset files every time you save. This gives you a file to go back to if you want to review an early iteration or if something happens to your working file. You can also set up Houdini to **AutoSave** in the **Edit > Preferences > Save and Load Options**. Just remember that all those backup files take up disk space and you will want to clear them from time to time.



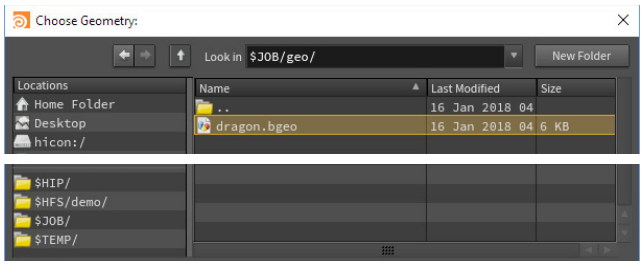
FILE SOP

When you import geometry into Houdini using **File > Import > Geometry** it puts down a **File** node at the geometry (SOP) level. This file maintains a connection to the file on disk and changes made to that file will also update in your Houdini scene. If you want to break this connection then you would need to lock the File node.



FILE DEPENDENCIES [$HIP/$JOB]

When you work with nodes that reference files on disk such as geometry or texture files, the path you use will determine what happens if you move the project directory to another computer or to the cloud. A direct path will break if you move the files therefore you should either use **$HIP** which uses the scene file as the "home base" for the path or **$JOB** which uses the project directory. You can use **Render > Preflight Scene** to check to make sure that your scene file is set up properly.



DISK SPACE MANAGEMENT

Large scene files, backup files and large simulations can take up **lots of disk space**. Be sure that you are not filling up too much space on your computer which could lead to instability issues. Try to use external drives to save out the largest files and leave the main disk on your computer with enough space to accomplish your day-to-day work.

INTEROPERABILITY

To import and export from Houdini, there are a wide variety of file formats that you can use. Here is a list of some of the main formats you will work with in a typical Houdini pipeline.

Houdini Files - Here are some file formats, other than .hip and .hda that work exclusively in Houdini.

.bgeo - This format saves geometry along with related attributes such as UVs, velocity and normals. Animations and simulations can be saved out as numbered bgeo files to save out motion. A bgeo.gz file is a compressed version of this format.

.sim - These files let you save out simulation data to cache the sim to disk. Some people use these files while others use .bgeos to cache sims.

.ifd - This is a scene description format that is created when rendering to Mantra. Typically these are created while rendering in Houdini but sometimes they are saved to disk to be rendered by Mantra directly.

.pic - This is an image file format that was used by Houdini in the past. It was replaced as the default format by the open source EXR.

.rat - This image format is ideal for texture maps being rendered in Mantra. All textures get converted to this format anyway so it speeds up rendering to convert to this format using Mplay.

Image Formats - These industry-standard formats are used to render out shots and for texture maps.

.exr - OpenEXR is a high dynamic-range (HDR) image file format developed by Industrial Light & Magic that is now the default format for saving out renderings from Houdini.

.jpg/.png - These formats are used to publish images to the web

.tga/.tif - These formats are often used to texture map video games.

Geometry Formats - When importing and exporting geometry, these formats are the most popular.

.usd - This is the format used in Solaris/LOPS and provides an open source interchange format for sharing with other applications.

.abc - Alembic is an open computer graphics interchange framework.

.fbx - This format owned by Autodesk is popular when exchanging data with game engines and other 3D applications. It can hold geometry, rigging, motion and shader information.

.obj - This is an simple geometry format originally created by Wavefront.

PREFLIGHT PANEL

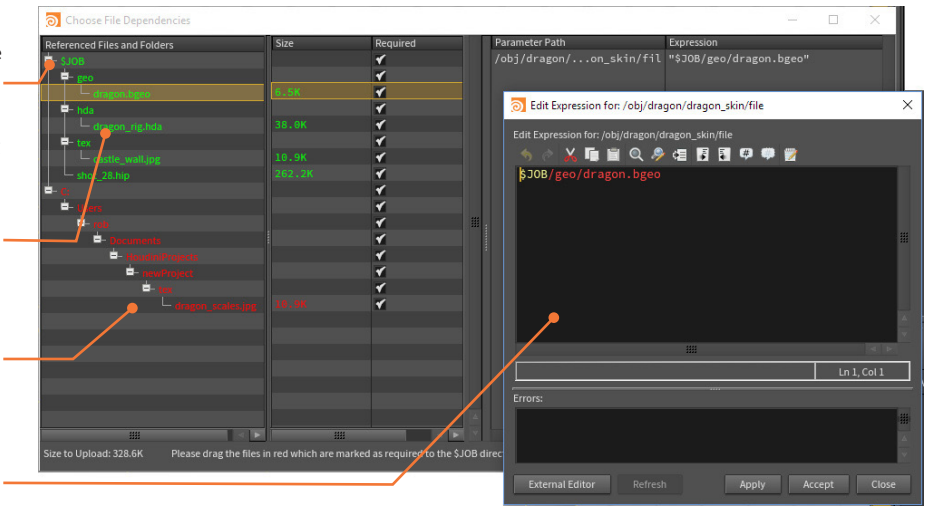
From the **Render** menu, select **Preflight Scene** to evaluate your scene setup.

Referenced Files - The preflight panel can either reference $HIP or $JOB when verifying file references for your scene file.

Greenlit Reference - If the reference is relative to either $HIP or $JOB then it will be displayed in green to indicate it is working.

Incorrect Reference - If a file reference is a direct path and not relative to $HIP or $JOB then it will be displayed in red and will need to get fixed before sharing your project with other artists or on the cloud.

Edit Expression - Click on any file name then on the right click on the expression to open up the edit expression window.

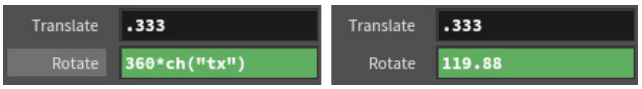


Expressions & Scripting

Houdini is a production-level solution which means that scripting will play an important role in your work. Artists can usually get by with simply writing expressions while technical directors will spend more time using these techniques. Houdini includes support for Hscript, Python and VEX.

HSCRIPT EXPRESSIONS

HScript is designed to be a fast and concise way to retrieve and manipulate information that can be used to write expressions. An expression is typically any value that is not either a simple string or number. This can be something as simple as a variable, a math equation or an expression function.



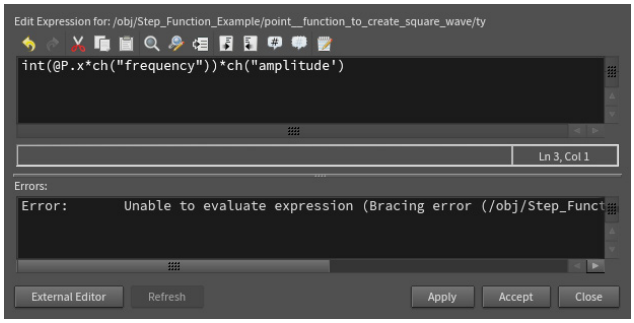
You can enter expressions directly into a parameter by simply typing into a field. When you press **Enter** the field highlights in green. You can click on the parameter name to toggle back and forth between the expression and the result of the expression.

If you are creating channel references then you can **RMB-click** on a parameter and choose **Copy Parameter** then go to the parameter you want to link it to and choose **Paste Relative References**.

You can also achieve this by RMB clicking on the second parameter then choosing **Reference > Scene Data**. This opens up a panel where you can choose data from other objects and nodes and an expression will be built for you. This method can even set expressions on multiple parameters.

EXPRESSION EDITOR

Depending on the complexity of your function, or the type of parameter, you may instead choose to use the **Expression Editor**. The expression editor can be opened by **RMB-clicking** on a parameter and selecting **Expression > Edit Expression,** or by placing the mouse over the parameter and pressing **Alt - E**.

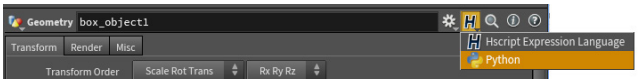


PYTHON

Python is a popular scripting language that is well known in the CG industry that supports integration and standardization. This makes it perfect for tool development.

Python in Houdini is built on the Houdini Object Model (HOM) which is an API that lets you get information from and control Houdini using the Python scripting language. In Python, the `hou` package is the top of a hierarchy of modules, functions, and classes that define the HOM. The `hou` module is automatically imported when you are writing expressions in the parameter editor and in the hython command-line shell.

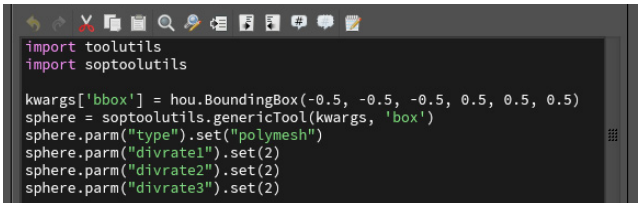
You can also use it to write expressions in Houdini. To do this, change the expression language option at the top of the parameter pane for the node.



There is also a **Python Shell Panel** which you can use to enter Python commands. You can also import the `hou` module into a regular Python shell to integrate Houdini into your existing Python-based scripts.

TOOL SHELVES

The shelf tools are also set up using Python. You can see this code by **RMB-clicking** on any shelf tool and choosing **Edit Tool.**

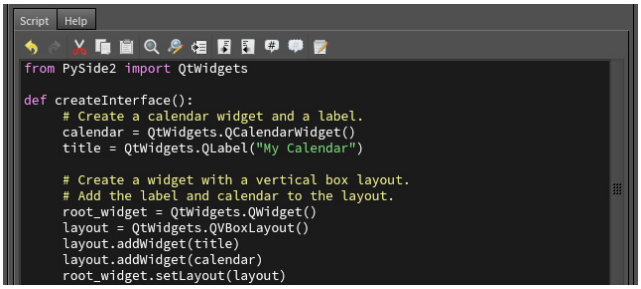


```
import toolutils
import soptoolutils

kwargs['bbox'] = hou.BoundingBox(-0.5, -0.5, -0.5, 0.5, 0.5, 0.5)
sphere = soptoolutils.genericTool(kwargs, 'box')
sphere.parm("type").set("polymesh")
sphere.parm("divrate1").set(2)
sphere.parm("divrate2").set(2)
sphere.parm("divrate3").set(2)
```

PYSIDE/PYQT

The **Python Panel Editor** pane lets you create, edit and delete PySide2 or PyQt5 interfaces. The editor also lets you manage the entries in the Python Panel interfaces menu as well as the entries in the Houdini pane tab menu. The panel comes with some sample code that you can try out for yourself.



```
Script  Help
from PySide2 import QtWidgets

def createInterface():
    # Create a calendar widget and a label.
    calendar = QtWidgets.QCalendarWidget()
    title = QtWidgets.QLabel("My Calendar")

    # Create a widget with a vertical box layout.
    # Add the label and calendar to the layout.
    root_widget = QtWidgets.QWidget()
    layout = QtWidgets.QVBoxLayout()
    layout.addWidget(title)
    layout.addWidget(calendar)
    root_widget.setLayout(layout)
```

PYTHON STATES

You can also write viewer states in Python that let you customize user interaction in the viewport for your node. You can use these to build more artist friendly interfaces for tools and you can refer to the documentation for more detailed info.

VEX

VEX is a high-performance expression language used in many places in Houdini, such as writing shaders. VEX evaluation is typically very efficient, giving performance close to compiled C/C++ code.

VEX is not an alternative to scripting, but rather a smaller, more efficient general purpose language for writing shaders and custom nodes. VEX is loosely based on the C language, but takes ideas from C++ as well as the RenderMan shading language.

VEX is used in several places in Houdini:

Modeling – The VEX SOP allows you to write a custom surface node that manipulates point attributes. This can move points around, adjust velocities, change colors. As well, you can group points or do many other useful tasks.

Rendering – Karma and Mantra use VEX for shading computation. This includes light, surface, displacement and fog shaders.

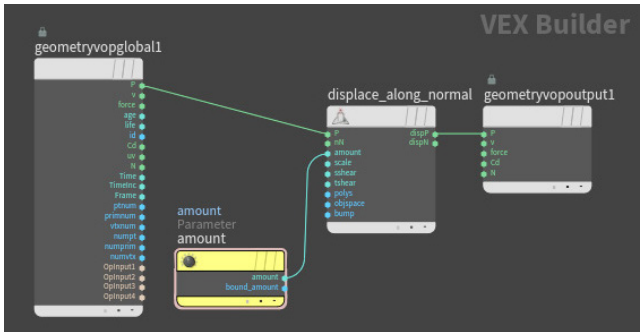
Compositing – The VEX Generator and VEX Filter COPs allows you to write complex custom COPs in VEX. The expressions evaluate very close to C/C++ speeds and run 1000's of times faster than the Pixel Expression COP.

CHOPs – The VEX CHOP lets you create custom CHOPs. The CHOP functions can manipulate arbitrary numbers of input channels and process channel data in arbitrary ways. In some cases, the VEX code can run faster than compiled C++ code.

Fur – Procedural fur behavior is implemented with VEX.

VOPS

If you want to use VEX but don't want to write the code then you can use the VOP context to use a node-based interface. You can do this in the SOP context using an **Attribute VOP** node that lets you dive in and use VOPs to create VEX code. You can take input geometry and manipulate it.

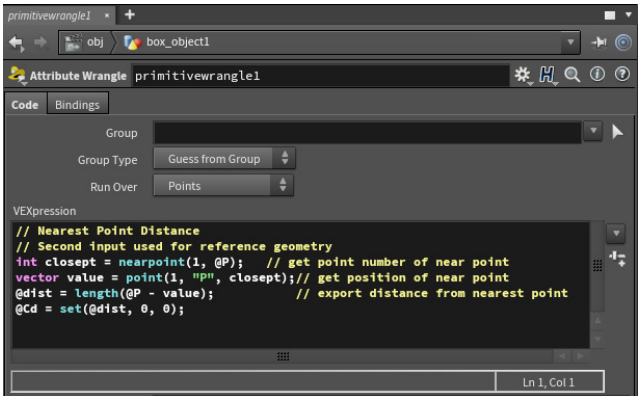


You can use parameter vops to build interface element such as float sliders that are then available at the SOP level. This way you can execute the VEX code without diving back down to the VOP level.

The VOPs context is designed to give artists an interactive way of creating VEX code. For people with a scripting background, it might make more sense to write the code directly into a Wrangle node.

WRANGLE NODES

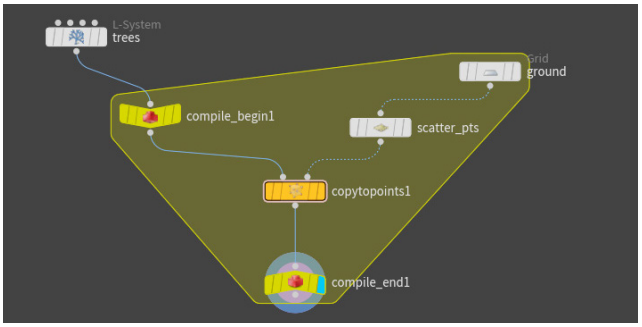
You can also use a wrangle node such as the **Attribute Wrangle** which provides a low-level node that lets coders who are familiar with VEX tweak attributes. There are also wrangle nodes for working with channels, volumes and deformations.



If you are interested in learning how to work with wrangle nodes, you should take a look at **Entagma.com** where you will find lots of great tutorials that generally take a more technical approach to creating content but with an artist's mindset.

COMPILE BLOCKS

In geometry networks [SOPs], you can put a part of the network inside a compiled block that makes it function as efficiently as if you had written code. This imposes a number of restrictions on how the network can work, but can potentially deliver big benefits in the right circumstances.



HOUDINI DEVELOPERS KIT | HDK

An even deeper way to work with Houdini is to use the HDK which is the same comprehensive set of C++ libraries that SideFX programmers use to develop the Houdini family of products. With the HDK, you can create plug-ins which customize different areas in the Houdini interface. Here are some examples of what you can do with the development kit:

- Add custom expression functions
- Add custom commands (hscript or HOM)
- Add custom operators (SOPs, COPs, DOPs, VOPs, ROPs, CHOPs, and even Objects)
- Add output nodes to support a non-standard Renderer
- Add custom lighting or atmospheric effects to the renderer

To learn more about working with the HDK, go to the SideFX website and choose **Support > Documentation > HDK**.

Tasks

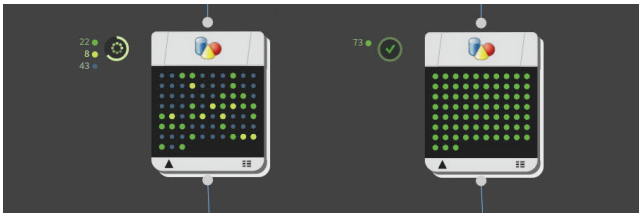
With the Task Operators or TOPs, you can organize and schedule tasks then distribute them intelligently to your compute farm. This allows for parallel processing of data while maintaining a dependency graph that shows how each task relates to proceeding tasks.

PROCEDURAL DEPENDENCY GRAPH

TOPs is a network type in Houdini built using the Procedural Dependency Graph, a technology which makes it possible to describe complex dependencies visually with nodes, then generate a set of actionable tasks which can be distributed to a compute farm with the help of a scheduler. Once you evaluate the results, it is possible to make changes to parts of the graph without re-cooking the whole network.

TOP NODES

Task or TOP nodes let you manage pipeline tasks with the ultimate goal of parallelizing the processing and distribution of each task. As a TOP node generates a task, the task is displayed as a dot. Once the task is cooked then new tasks can be executed on this node and on children TOP nodes.



SCHEDULERS

Scheduler nodes take tasks that have met required dependencies and assigns compute resources. As each task is completed, the scheduler informs the task graph, which in turn informs the PDG graph to move on to the next available task. PDG supports industry-standard schedulers such as HQueue, Deadline, Tractor, or any scheduler plugged in with Python.

COOKING TOP NODES

Once you wire together a task graph, you will want to cook the nodes. You can either cook a node in the middle of the graph or cook the output node at the end of the chain.

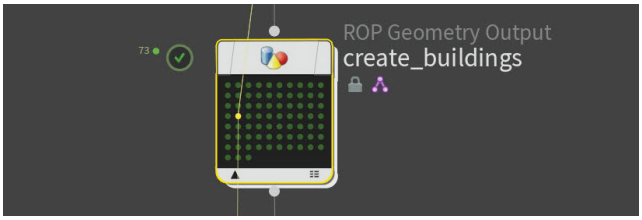
- Cook Selected Node Shift-G
- Dirty and Cook Selected Node Shift-V

Use the Task Bar to monitor your progress. On the TOP nodes, you can RMB-click on a task dot and choose to Cook or Dirty the task. When you dirty a task, it means that if you recook the network those tasks will be recomputed. Clean tasks will not be recooked which is one of the benefits of TOPs because you don't have to redo work that is already completed.

DEPENDENCIES

When you click on a task dot in the graph, you will see a thin line that connects to upstream tasks it is dependent on and downstream tasks that depend on it.

If there are changes upstream then tasks may be automatically dirtied and this will in turn dirty tasks downstream where there are dependencies. This process is an important part of how PDG graphs work as an effective pipeline tool.



TOP NODE

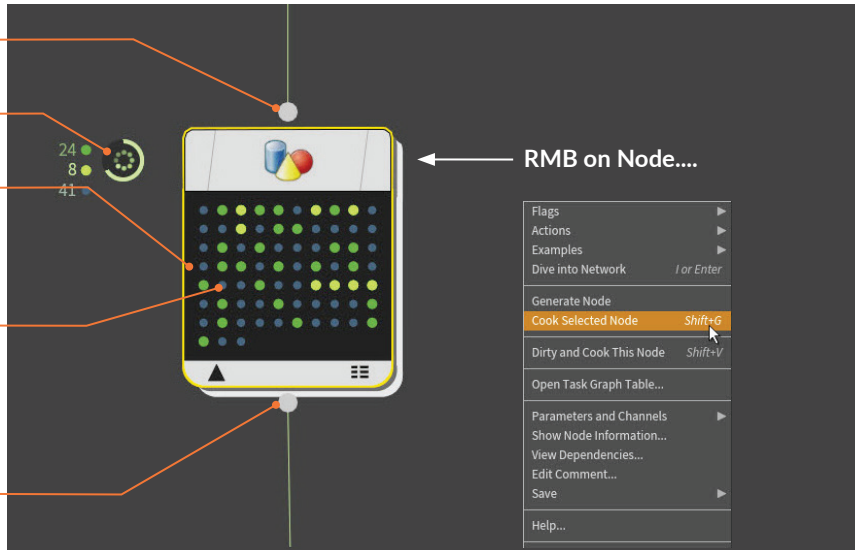
Input - The node takes the information feeding into the Input and breaks it into one task for every piece of data.

Progress Wheel - The progress wheel shows you how many tasks are completed, how many are in progress and how many are in the queue.

TOP Node - This is the node that is currently being cooked. It contains the instructions for the tasks being executed. You can RMB-click on the node for a menu of supporting actions.

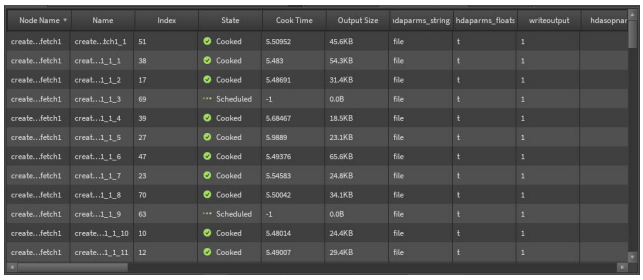
Tasks - Each task is represented by a small dot. The coloring indicates their current status and you can RMB-click on a task dot to learn more about that one part of the graph.

Output - Once tasks are completed, the output passes the results on to the next node even if other tasks on this node are still active.



TASK GRAPH TABLE

If you **RMB-click** on a node, you can choose to **Open Task Graph Table**. This gives you an itemized list of tasks along with information such as index, state, cook time and priority. Clicking on items in this window will highlight the task dot on the node in the Network view.

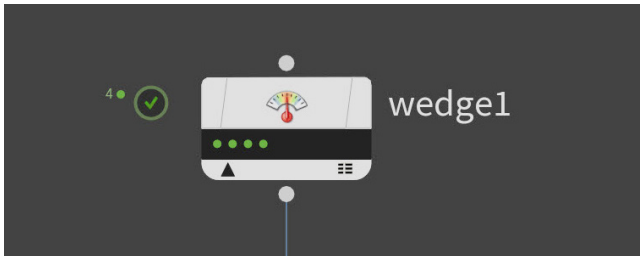


IMPORT/EXPORT DATA

To get data into the TOP graph, there are a number of different options giving you access to geometry, images, scripts and other kinds of data. Houdini Digital Assets can be used to apply procedural networks or you can connect with other parts of Houdini to import and export data.

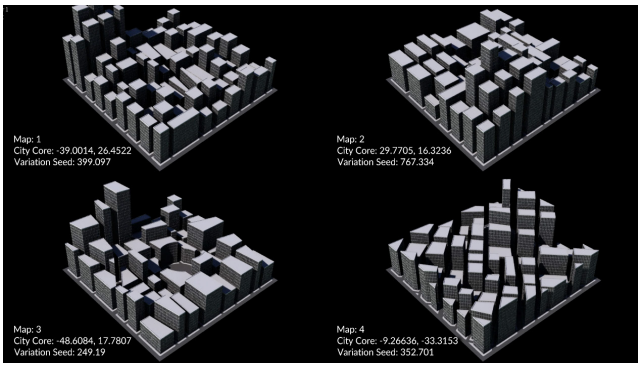
WEDGE NODE

A key workflow in PDG is wedging that lets you quickly create multiple iterations of a design. You can then process all of the different options through the TOP graph then collect them at the end for final output.



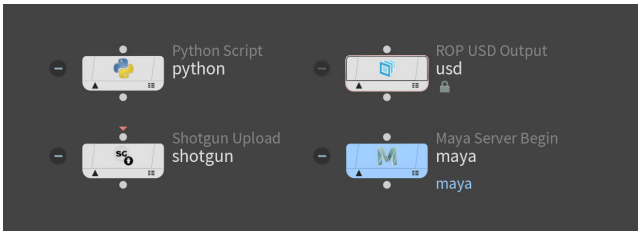
OUTPUT IMAGE MOSAICS AND MOVIES

In TOPS, you can interface with **ImageMagik** to create a contact sheet that can be used to evaluate design iterations to make the best choice or to generate prop variations to richly populate your scene. You can use an overlay to pull info from the network to help you make the best decision.



INTEGRATIONS WITH OTHER APPS

TOPs includes nodes for working with other applications such as Shotgun or Autodesk Maya. This allows your network to extend beyond Houdini to help with all parts of the pipeline.



PILOT PDG APPLICATION

While TOP networks can be set up and executed from within Houdini, wranglers who are managing the farm or pipeline TDs who are exclusively creating TOP networks can use **PilotPDG**. Tasks that are Houdini-related will call on Houdini Engine to work non-graphically to complete the task.

TOP NETWORK

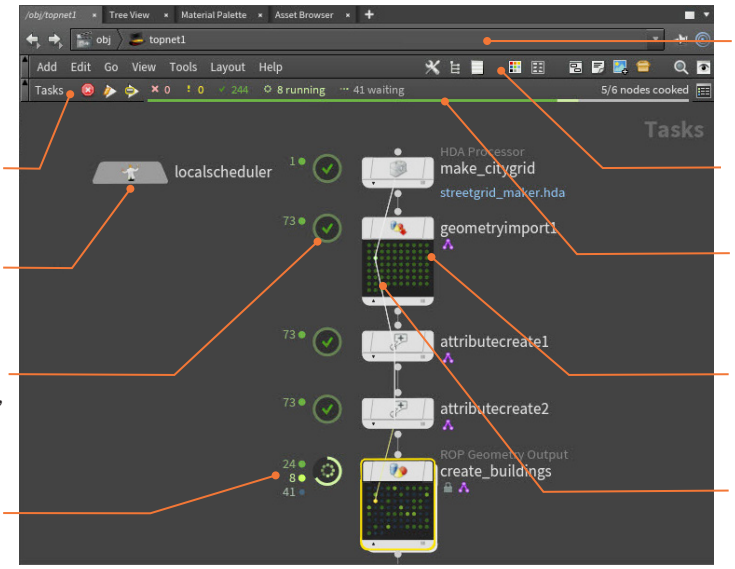
This network type lets you manage and view the network being processed.

Task Bar- The task bar lets you start and stop a network and monitor its progress.

Scheduler- The scheduler node determines where your data is being processed and how many nodes are participating.

Completed Tasks - When a node is finished processing all its tasks, a check mark appears.

In Progress Tasks - While in progress, you can see which tasks still need to be completed.



Network Path- This shows you the path to the TOP network where the graph is set up.

TOP Menu- This menu includes a range of options for organizing and processing a TOP network.

Progress Bar - This bar lets you see the progress of the overall network tasks.

TOP nodes - These nodes are where specific commands are turned into tasks and sent by the scheduler to be completed.

Dependency Line - You can click on a task to see how it connects to other tasks in the network.

HOUDINI DIGITAL ASSETS
Procedural Tool Building

Networks of nodes give Houdini its procedural nature and define a recipe that can be applied over and over. Houdini Digital Assets let you wrap up these networks to create custom tools and smart assets. These artist-built tools can be used repeatedly to increase productivity across your studio.

One of the things that Houdini's node-based workflow is great at is allowing artists to avoid repetitive steps and to generate multiple iterations by making changes to an existing network of nodes. This lets you achieve results that are unique without starting the whole process from scratch.

Houdini Digital Assets take this one step further by letting you encapsulate a network or collection of networks into a single node with parameters that have been promoted to the top level. This node is then saved to disk which creates a shareable file that other artists can load into their scenes.

ARTIST BUILT TOOLS

The process of creating a Houdini Digital Asset works with the interactive tools in Houdini. You build a high level interface by dragging parameters from nodes to an asset properties panel which allows you to create custom tools without writing any code. This means that technical artists can build custom tools then deploy them quickly to colleagues.

A Houdini Digital Asset might be a procedural prop such as a staircase or a piece of furniture, a visual effect such as an explosion, or a more generalized tool such as a populate tool for scattering objects over a surface. Whether you are creating content specifically for your current project or building a larger toolset for all your projects, your artists can build a collection of Houdini Digital Assets to meet your production needs.

PIPELINE FRIENDLY

When a Houdini Digital Asset is loaded into a scene file, it references the **.hda** file on disk. This means that changes made to the asset will be picked up automatically by everyone who is referencing that file.

This makes it very easy to deploy updates throughout your pipeline. Now artists can point to a single asset on disk knowing that once it gets updated with the most current iteration, they will immediately have access.

Houdini Digital Asset files can also hold more than just the asset definition. You can store images, geometry files and scripts that are used by the asset. This ensures that all the relevant parts are available when other people work with the asset.

CONTENT LIBRARY & ORBOLT

The **Content Library** is an online asset repository which hosts 2D and 3D assets, from complete scene files, to fully-rigged props, to render-ready visual effects, animatable characters, game assets and more. Go to **Get > Content Library** on the SideFX website to access it.

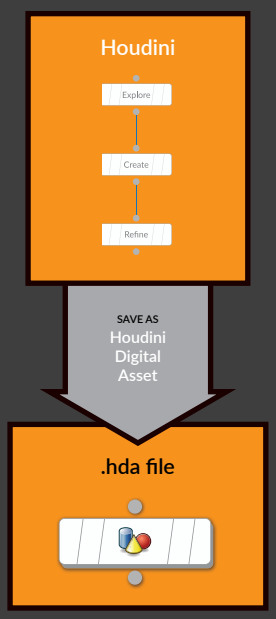
Orbolt an online asset marketplace that offers a wide variety of digital assets. There is a panel in Houdini where Orbolt assets you download or purchase can be stored and made available as you work.

CREATING DIGITAL ASSETS

1 Create nodes and networks in Houdini

2 Package up the networks to be saved out as a Houdini Digital Asset [.hda] file that can be shared with other artists.

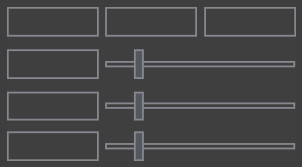
3 Build an interface for your asset by promoting parameters and handles to the top level of the asset.



4 Load the .hda file back into Houdini to use the asset.

Only those parameters promoted to the asset level can be used. All others are locked.

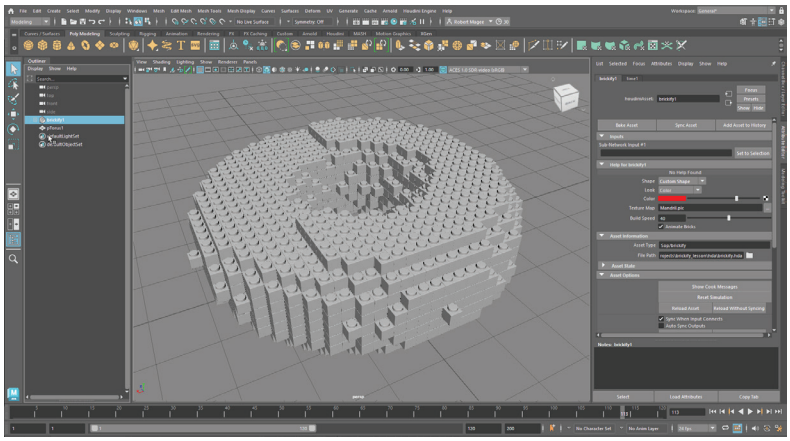
You can use the asset in any number of Houdini scenes. If you make changes to the HDA file then all other assets can be easily synced to the changes.



HOUDINI ENGINE
Sharing with other Apps

Houdini Engine brings a procedural node-based approach to your favorite app. This technology lets you share Houdini Digital Assets with colleagues who can load them directly into 3D apps such as Autodesk® Maya® or 3DS MAX® or into game editors such as Unity® or Unreal.®

The benefits of Houdini Digital Assets can be experienced by artists using other applications thanks to the Houdini Engine plug-ins. Created with the Houdini Engine API, these plug-ins allow host applications to load .hda files and all of the handles and controls. When parameters are set on the asset, Houdini works "under-the-hood" to cook the nodes and networks then deliver the results back to the host.



A Houdini Digital Asset loaded into Autodesk Maya using the Houdini Engine

HOUDINI ENGINE API

This is made possible by the Houdini API which is used to create plug-ins for host applications. HAPI is a flat and small API that is easy to learn and is available on **github** for developers that want to create their own proprietary plug-in.

HOUDINI ENGINE PLUG-INS

There are a number of Houdini Engine plug-ins that artists can access either through the Houdini installer or online. These have been production tested and can be used confidently by artists and studios.

Each of the plug-ins are designed to create a bridge between the features in a typical Houdini asset and the nature of the host application. For instance a cloud asset that use volumes would work fine in Maya but would not make sense in Unity or Unreal where volumes are not supported.

Plug-ins that work with **FREE Houdini Engine for Unity/Unreal** or **FREE Houdini Engine Indie** licenses:

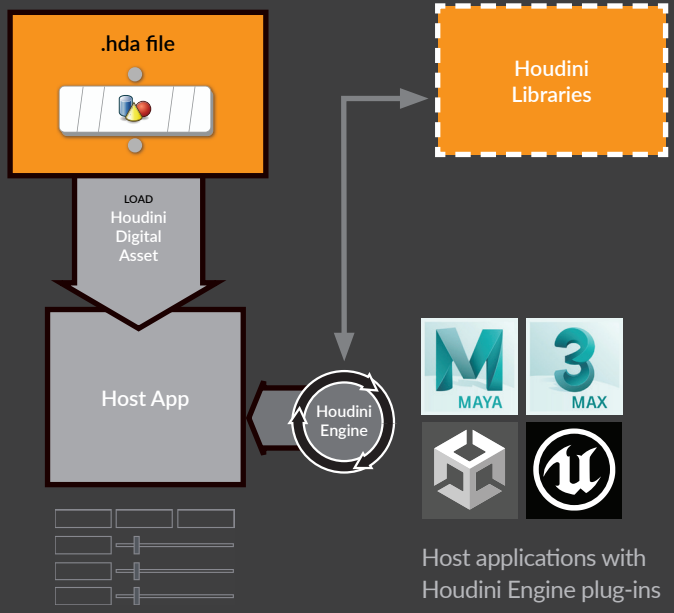
- **Unreal**
- **Unity**

Plug-ins that work with **Houdini Engine** licenses or **FREE Houdini Engine Indie** licenses:

- **Autodesk Maya**
- **Autodesk 3DS Max**
- Proprietary Plug-ins

HOUDINI ENGINE PIPELINE

1. Load the .hda file into a host application using the Houdini Engine plug-in.
2. The Host Application accepts the asset and interfaces with the Houdini Engine.
3. The Houdini Engine calls on the Houdini library files to "cook" the nodes and network inside the asset.
4. When an asset is loaded or a parameter is changed then the Engine grabs the Houdini libraries, cooks the nodes, then delivers the results back to the Host.



Host applications with Houdini Engine plug-ins

FILM & TV PIPELINE
Animation and VFX

Whether you are creating live action plates enhanced with visual effects or full CG shots, the ultimate goal of Film and TV projects is moving pictures. These pictures are created using assets such as characters, sets and effects which all come together in a final composition.

Houdini is a full featured package that contributes to all stages of a Film & TV pipeline. From modeling to rendering to animation and final compositing, Houdini has procedural tools that support your creative process. Over the years, VFX is one area where Houdini is known as an industry standard. SideFX has been honored with several Scientific and Technical Achievement awards including an ACADEMY AWARD OF MERIT Oscar.

Other areas such as procedural modeling, lighting or character work continue to get stronger to the point where a growing request from studios is more skilled Houdini artists.

HOUDINI CORE / HOUDINI FX

There are two commercial versions of Houdini that you use in your pipeline. Houdini Core covers all of Houdini's tools except for DOPS, and Houdini FX has a full toolset. Scenes and VFX created in Houdini FX can be staged, animated, lit and rendered in Houdini Core. This gives you a robust pipeline with Houdini FX licenses for your FX artists and Houdini Core licenses for everyone else.

One solution is for senior technical directors to use Houdini FX to solve a particular production challenge then wrap up the resulting nodes and networks into Houdini Digital Assets. An artist-friendly UI is then built to support the animators and VFX artists who can then use the more cost effective Houdini Core to execute shots.

INTEROPERABILITY

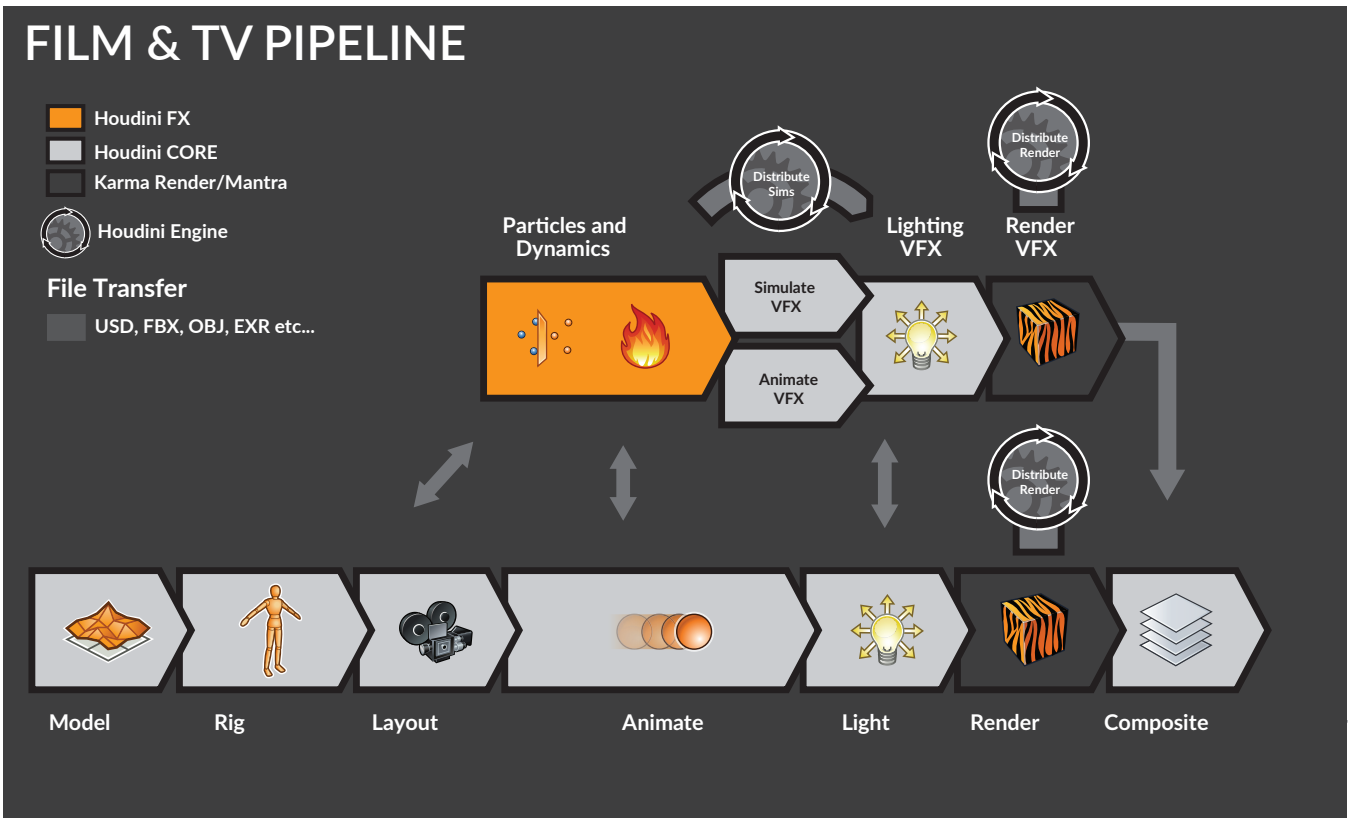
Most studios are equipped with a variety of 3D applications to each handle a different part of the pipeline. Houdini has a lot of strong interoperability tools to allow for this interchange of data. Whether they are using USD, Alembic, FBX or EXR, your artists can easily work back and forth with a wide variety of DCC applications. They can also use the Houdini Engine plug-ins to bring the Houdini Digital Assets into other applications, such as Autodesk® Maya® or 3DS MAX,® while maintaining the asset's procedural controls.

Smaller studios may want to avoid extensive file exchange, especially with tight deadlines, therefore Houdini provides a full featured procedural "pipeline-in-a-box" that can take you through all of the stages under one roof.

DISTRIBUTED RENDERS AND SIMS

Rendering images and simulating VFX can be time consuming, especially as you aspire towards photorealistic results. For this reason, Houdini lets you distribute both rendering and simulation tasks to a compute farm using **Houdini Engine** in **Batch** mode.

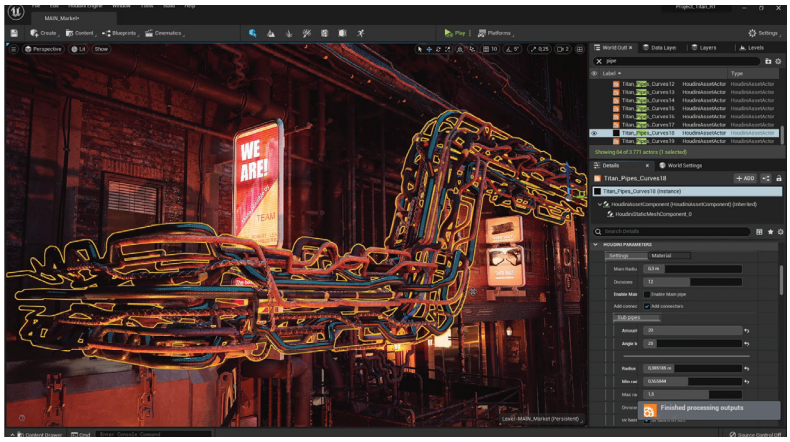
Distributed simulations allow you to work faster or to handle effects that would max out the memory on any one computer. By slicing the sim and distributing it, memory is managed without compromising the final result. Studios should definitely consider using **Houdini Engine** to simulate on the farm.



GAMEDEV & VR PIPELINE
Interactive Experiences

In Video Game and Virtual Reality projects, the main focus is creating interactive 3D worlds built using content that is highly optimized for a smooth gameplay experience. This creates a different kind of pipeline compared to rendered out game cinematics which are more like film.

At the core of a games pipeline is a game engine like Unreal or Unity. The engine is where the game art and the game interactions are put together to create a playable experience. Houdini can be used by game artists to create terrain, design and populate levels, build procedural models, build and animate characters and create Realtime FX such as fire, fluids and destruction.



Houdini Digital Assets loaded into Unreal using the Houdini Engine

EXPORTING TO GAME ENGINES

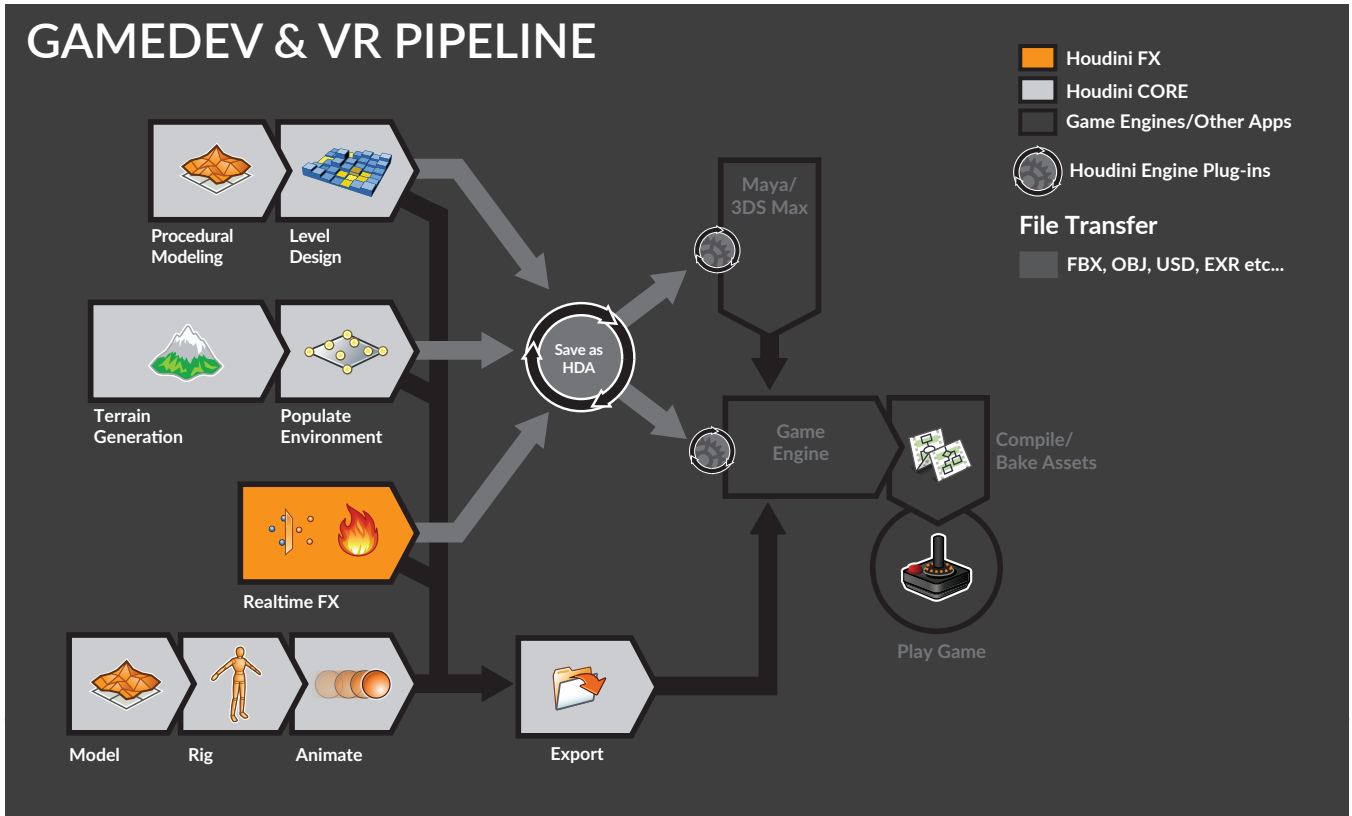
There are two ways of getting content from Houdini to a game engine. The traditional approach is to export out to a format like FBX or OBJ and import this into the engine. You would create procedural systems in Houdini then flatten out the results.

The second approach is to create **Houdini Digital Assets** and load these into the game engines using the **Houdini Engine plug-ins** for Unreal and Unity. These assets import into the game editor with their parameters and controls intact. You can therefore make changes inside the game editor and the Houdini Engine works in the background to update the artwork.

This proceduralism is available to game artists inside the editor then when the game is compiled the artwork is baked down. The Houdini Engine is not a runtime solution and you cannot access it as part of the gameplay.

REALTIME FX

Houdini is known for VFX and it is a great tool for creating FX for games. But these FX need to be optimized using techniques such as texture sheets, flowmaps and vertex animation textures. This way the footprint for the effect is as light as possible and does not take away from the frames per second of the game. The SideFX Labs Tools mentioned earlier in this document have been designed to support these kinds of workflows.



Products & Licensing

As you begin working with Houdini, it is useful to understand what Houdini products are available for you to work with. Whether you are a large studio, small studio or a team of indies just getting started, there are different Houdini products to suit your needs. There are also versions of Houdini for school labs and for students who want to learn for free.

COMMERCIAL LICENSES

Houdini Core – Designed for modelers, lighters, character riggers, animators and game artists, Houdini Core also includes features such as compositing and motion editing. Scenes created in Houdini FX can be opened and rendered in Houdini Core which makes it an ideal lighting tool for your VFX.

Houdini FX – Houdini FX has all the tools found in Houdini Core but adds particle and dynamic simulation tools. With its procedural workflow, Houdini FX lets you create fluids, Pyro FX, grains, cloth, hair & fur, crowds and soft body effects.

Houdini Engine – Houdini Engine gives you command-line access to run in batch mode to batch process renderings and distributed dynamic simulations. Houdini Engine also lets you load Houdini Digital Assets into other digital content creation applications, such as Autodesk® Maya,® Autodesk® 3DS MAX,® Unity® or Unreal.®

INDIE LICENSES

Houdini Indie – Houdini Indie makes all of Houdini's animation and VFX tools available under a limited commercial [less than $100K USD] license to animators and game makers who want to use Houdini during the incubation stage of their business.

Houdini Engine Indie – Your Houdini Engine Indie license can be used to run Houdini Indie in batch mode or to load Houdini Digital Assets into other content creation apps.

LEARNING LICENSES

Houdini Education – Houdini Education is a full-featured version of Houdini FX designed for use by schools and training centers as well as students. Houdini Education will also open files created by using Houdini Apprentice.

Houdini Apprentice – Houdini Apprentice is a free version of Houdini FX which can be used by students, artists and hobbyists to create personal non-commercial projects. With Houdini Apprentice, you have access to virtually all of the features of the award-winning Houdini FX to develop your skills and work on personal projects. Apprentice lets you save to disk and render out with a word mark.

Note: *Scene files and Assets created in Indie, Apprentice or Education cannot be used in commercial Houdini. The file formats are different and the EULA [End User License Agreement] prevents you from sharing files between different license types.*

LICENSE TYPES

Workstation [Node-Locked] – This license type can be used on a single computer and can only be accessed either from a local server or from SideFX.com.

Local/Global Access [Floating] – These licenses can be set up on a server and shared with a team of artists. When an artist starts Houdini a license is checked out of the server as long as there is one available. Local licenses are designed for a single studio and global licenses are for sharing between studios in different locations.

INSTALLING LICENSES

Once you have acquired a license, you will install it by opening up the **Houdini License Administrator [hkey]** application. From there you can choose **File > Install Licenses**. You will be asked for a log in and a password that will match the ones you set up on the SideFX.com website.

You can now use sidefx.com as the license server instead of installing locally. To use **Login licensing**, you need to be constantly logged in with your sidefx account. You can login using different computers but only one of them can be used at a time. This approach is ideal for Indie and Education users.

Local and Global Access licenses can be installed using this method on a central server. You will then need to make that server available to anyone who needs access to the licenses.

You can also view your licenses on the SideFX.com website by clicking on your avatar in the top right and choosing **Services**. You can then click on the **Manage Licenses** link.

ANNUAL UPGRADE PLAN

For visual effects studios, games studios and 3D artists who want to maximize their investment in Houdini, the Annual Upgrade Plan provides key advantages such as production-level technical support and access to full and dot releases containing the latest software enhancements as well as daily builds containing bug fixes.

SIDEFX SUPPORT

All customers including Apprentice customers can contact SideFX using our email support system to discuss installation and licensing issues. After that, only Annual Upgrade Plan and Commercial Rental Customers may contact our support team to discuss more in depth production issues.

Our Support Specialists can be contacted directly via **support@sidefx.com** . Be sure to include the following information in your email:

- *Your Operating system [Windows XP, etc.]*
- *Version and Build Number of Houdini*
- *Summary of the installation issue and a diagnostic file if you are having a licensing issue.*

To learn about support programs visit **SideFX.com/support**.

Comparison Chart

		COMMERCIAL		INDIE	LEARNING	
	PRODUCT	HOUDINI FX	HOUDINI CORE	HOUDINI INDIE	EDUCATION	APPRENTICE
	INTENDED USER	Studios | Commercial Artists		Indies | Freelancers	Schools | Students	Hobbyists
	PRICING	Visit SideFX.com		$269 USD per year	$75 USD per year	FREE
	Operating System	Windows, LINUX, Mac OSX				
FEATURES	Modeling	✓	✓	✓	✓	✓
	Character	✓	✓	✓	✓	✓
	Animation	✓	✓	✓	✓	✓
	Solaris: Layout Tools	✓	✓	✓	✓	✓
	Solaris: Lookdev and Lighting	✓	✓	✓	✓	✓
	Karma/Mantra Rendering	✓	✓	✓	✓	✓
	Terrain	✓	✓	✓	✓	✓
	Compositing	✓	✓	✓	✓	✓
	Volumes	✓	✓	✓	✓	✓
	Pyro FX	✓	Simple Fireball	✓	✓	✓
	Fluids	✓	Simple Flip	✓	✓	✓
	Rigid Bodies	✓	Simple Fracture	✓	✓	✓
	Particles	✓	-	✓	✓	✓
	Vellum Cloth	✓	Simple Cloth	✓	✓	✓
	Wire Dynamics	✓	-	✓	✓	✓
	Crowds	✓	-	✓	✓	✓
	LICENSING	Commercial		Limited Commercial	Non-Commercial	
	Workstation [Node-Locked]	✓	✓	✓	-	✓
	Local/Global Access [Floating]	✓	✓	-	✓	-
	USER INTERFACE					
	Houdini GUI Access	✓	✓	✓	✓	✓
	Command Line Access	✓	✓	✓	✓	✓
	GUI Watermark	-	-	Unobtrusive	Unobtrusive	Unobtrusive
	Plug-in Support	✓	✓	✓	✓	✓
	HOUDINI ENGINE					
	Houdini Engine Plug-ins	✓	✓	✓	✓	No
	Create Assets for Engine	✓	✓	✓	✓	For Education Licenses
	Create Assets for Orbolt	✓	✓	✓	✓	✓
	RENDERING					
	Karma Tokens	5 / 10*	5 / 10*	1	10	1
	Mantra Tokens	Unlimited	Unlimited	1	10	1
	3rd Party Rendering	✓	✓	✓	✓	No
	Render Watermark	-	-	-	-	✓
	Resolution	Unlimited	Unlimited	Unlimited	Unlimited	1280x720
FORMATS	SCENE					
	.hip	✓	✓	.hipalc	.hipanc	.hipanc
	.hda	✓	✓	.hdalc	.hdanc	.hdanc
	GEOMETRY					
	USD	✓	✓	✓	✓	.usdnc
	FBX	✓	✓	✓	✓	IMPORT
	Alembic	✓	✓	✓	✓	IMPORT
	.bgeo	✓	✓	✓	✓	✓
	IMAGES					
	.pic	✓	✓	.piclc	✓	.picnc
	.exr	✓	✓	✓	✓	watermarked
	.tif	✓	✓	✓	✓	watermarked
	.png/.jpg	✓	✓	✓	✓	watermarked

* Commercial Workstation Licenses come with 5 Karma Tokens and Local and Global Access Licenses come with 10 Karma tokens

HOUDINI FOUNDATIONS

MODEL, RENDER, ANIMATE

Welcome to Houdini. In this lesson you will start from scratch to model, render, animate, and simulate a soccer ball (also known as a football in many parts of the world). You will create a classic bouncing ball animation using the principles of squash and stretch, apply textures and materials, add lights and cameras, and explore the use of dynamics to simulate a group of soccer balls.

These tasks will introduce you to many different parts of Houdini as you create your first Houdini scene, explore the interface and discover some of its most important tools. You will learn how to work interactively in the **Scene View** and how to use the **Network View** to manage your nodes as you refine your model and build your animation rig. You will also set up materials and textures on the **Solaris Stage** then you will render using Houdini's built-in renderer **Karma,** and finally create a **Rigid Body Simulation**.

LESSON GOAL

Model, Render, Animate and Simulate a soccer ball using Houdini's procedural node-based workflow

WHAT YOU WILL LEARN

- How to work with the **View Tools**
- How to use **Shelves**, **Radial Menus** and the **Tab** key
- How to create **Geometry**
- How to work with **Nodes and Networks**
- How to set up **Custom Attributes** and a **For-Each Loop**
- How to set up **Materials** and **Texture UVs**
- How to **Layout** a shot and render with **Karma**
- How to **Set Keyframes** and add **Motion FX**
- How to use **Rigid Body Dynamics**

LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 4.0 | July 2022

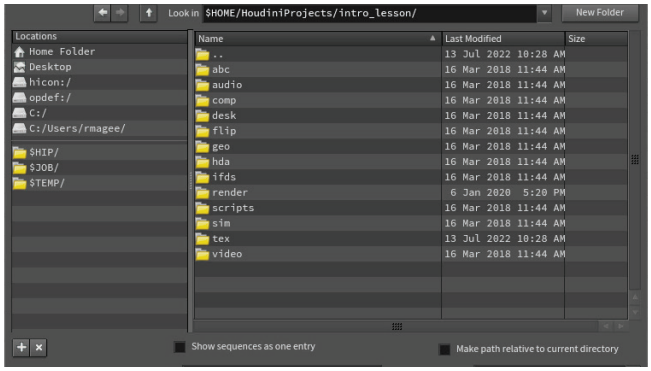
PART ONE

Explore the Houdini UI

To get started, it is important to learn how to work with the Houdini workspace and the three panes you will use the most. The **Viewport** lets you create objects interactively, the **Parameter Pane** lets you edit node properties and the **Network Editor** lets you work directly with the node networks.

PROJECT FILES

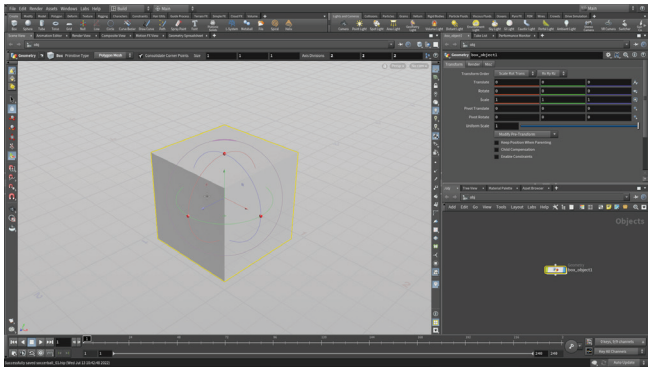
Go to the **soccerball tutorial** page on **SideFX.com**, where you likely got this document, to download the *intro_lesson* directory. Put it into the **Houdini Projects** directory which you can find in either the **home** directory or the **documents** directory.



01

Select **File > Set Project.** Find the *intro_lesson* directory that you downloaded earlier and press **Accept**. This makes this project directory and its sub folders the place for all the files associated with this shot.

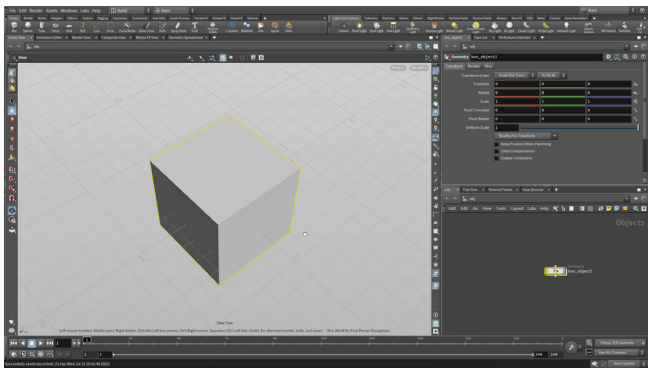
Select **File > Save As...** You should be looking into the new *intro_lesson* directory. Set the file name to *soccerball_01.hip* (or *football_01.hip* if you would prefer) and click **Accept** to save.



02

In the viewport, **press c** to bring up a radial menu. From this menu, choose **Create > Geometry > Box**. Your cursor now shows the outline of a box waiting to be placed in the scene. **Press Enter** to place it at the origin.

This creates a box in the Scene view, adds a node in the Network editor and shows the object parameters in the Parameter pane. As you work through this project, you will touch on all of these interface elements.



03

You can now explore the **View** tool in Houdini. Press the following hotkeys:

- **Tumble** — Spacebar or Alt[Opt] - LMB click-drag
- **Pan** — Spacebar or Alt[Opt] - MMB click-drag
- **Dolly** — Spacebar or Alt[Opt] - RMB click-drag

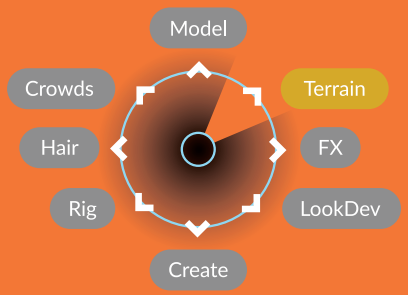
In some cases, you will want to home in to get your bearings. There are some hotkeys for that as well:

- **Home Grid** — Spacebar -H
- **Home All** — Spacebar - A
- **Home Selected** — Spacebar - G

RADIAL MENUS

One way to access tools in Houdini is radial menus which you can access using the **X**, **C** and **V** hotkeys. Each of these brings up a radial menu with lots of options for you to choose from. The main focus of each menu is as follows:

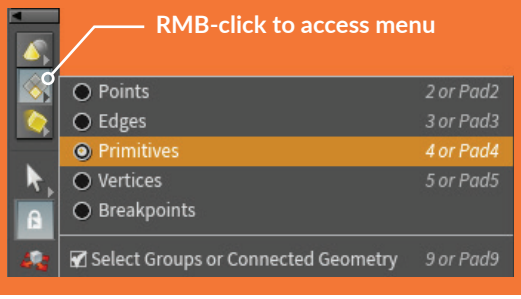
Snapping	**X**
Main (or Custom)	**C**
View	**V**

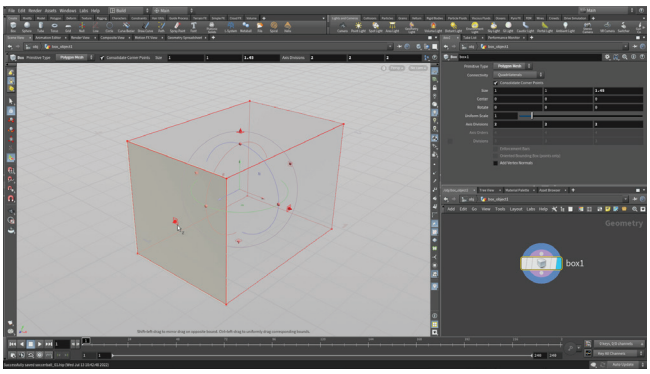


SELECTION HOTKEYS

If you are using the **Select, Move, Rotate, Scale** or **Handles** tools, the following hotkeys will determine your selection mode as well as which level you will be working at.

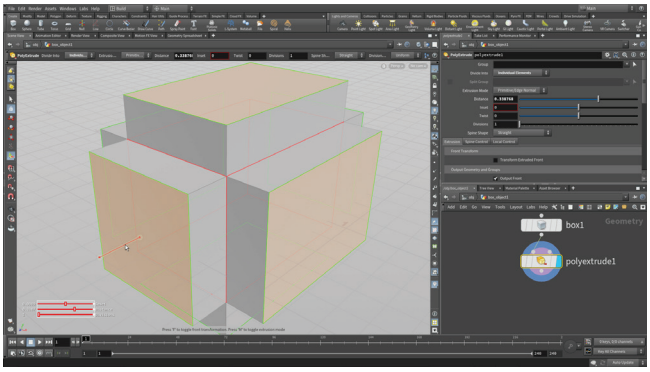
Objects	Object Level	1
Points	Geometry Level	2
Edges	Geometry Level	3
Primitives/Faces	Geometry Level	4
Vertices	Geometry Level	5





04 With the object selected, **press i** to go to its geometry level. Use the **Shift** key to drag on handles to make it longer along z axis around the origin.

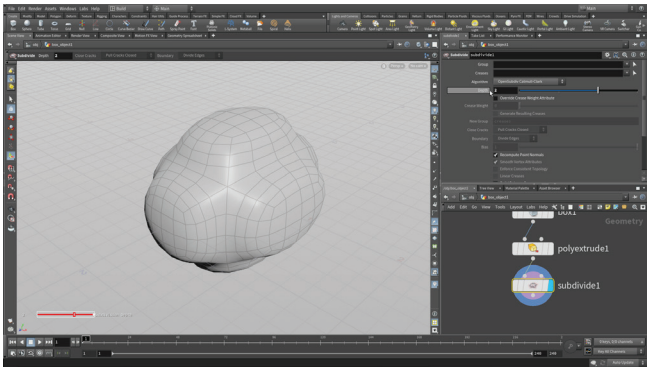
When an object is created in Houdini, there is an **Object level** which is where you manage the transformations of the object and a **Geometry level** where you define its shape. Pressing **i** brought you down into the geometry level of this object. You can also get there by double-clicking on the object node in the Network editor. Later, to get back to the Object level, you will **press U**.



05 **Press S** to go to the **Select tool** then **4** to access Primitive selection. **Press n** to select all then **press c** to bring up the radial menu and choose **Model > Polygons > Poly Extrude**.

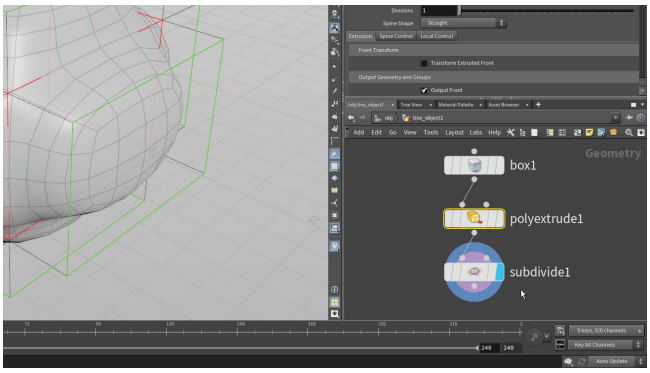
In the either the **Operation Control Bar** at the top of the Scene view or the **Parameters** pane, set **Divide Into** to **Individual Elements** and use the handle to set the **Distance** to around **0.4**. This extrudes the faces along the normals of each primitive.

You can see that there are now two nodes in the Network view. Each step you take in Houdini creates a node that you can work with to refine your scene.



06 Press **n** to select all of the new faces and press **Tab** and begin typing **sub...** then select **Subdivide** from the list. The **Tab** key is another way to access tools in Houdini. Typing the tool name lets you focus the list making it easier to find what you want without navigating the submenus.

In the Parameter pane, set **Depth** to **2**. This subdivides the geometry to create more polygons. Houdini also has a subdivision display option at the Object level which you can use to see subdivisions without actually adding any geometry, but in this case you do want to create more polygons.



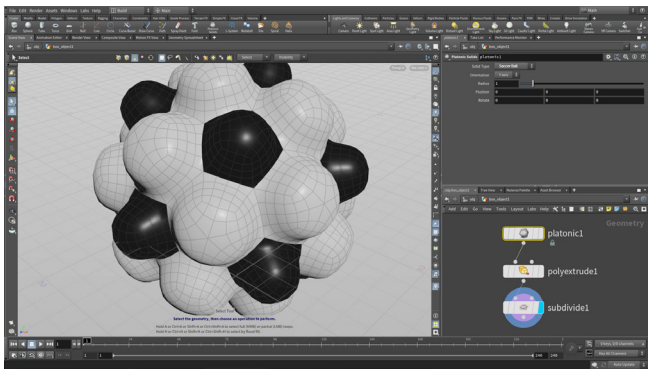
07 Select the different nodes in the chain. The handles for each of the nodes appear as you select them but the display remains on the final shape. Set the **Display Flag** on each of the nodes to change which node is the display node. You can also try some of the other flags such as **Bypass** or **Template**. **Wiggle** the *polyextrude* node out of the network then drop it back in.

At the end, return everything to normal and set the **Display flag** on the *subdivide* node. This is very important. The Display flag determines what you will see at the object level. **Always check to make sure you have the right display flag set!**

PART TWO

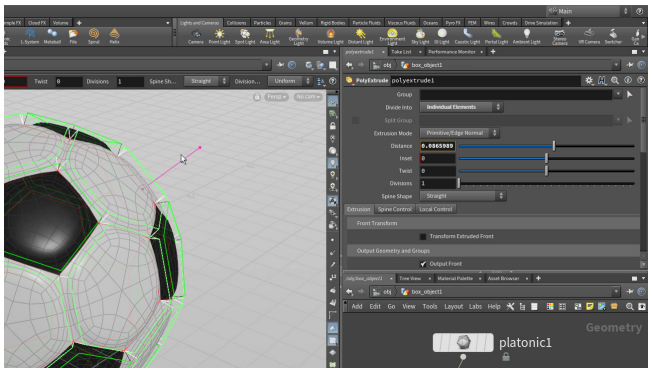
Create a Soccerball

You are now going to replace the box with a soccerball shaped platonic shape. Using Houdini's procedural approach, you can replace the box node with a platonic solid node. From there you will adjust the other nodes to make it look like a soccerball. This ability to swap out input nodes lets you prototype networks with simple geometry for added flexibility.



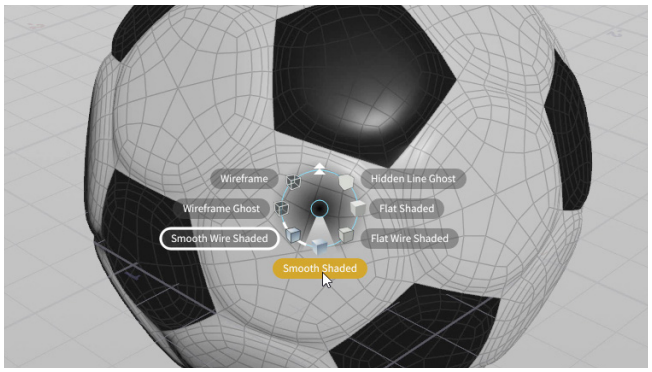
01 In the **Network editor**, use the **Tab** key to add a **Platonic Solids** node to the network. Click to place it down near the top of the chain. Wire the *platonic* node into the *polyextrude* node. In the parameter pane, set **Solid Type** to **Soccer Ball**. Select and delete the *box* node.

Because of Houdini's procedural nature, it is often possible to replace an input node and have the whole network function properly. This gives you flexibility as you work and if you don't like the results after the change then you can always wire back the original shape.



02 Select the *polyextrude* node. Make sure the **Handle** tool is active then use the handle in the viewport to set a smaller **Distance**. You can also set the parameter value in the Parameter pane. This creates a better look for the soccerball. Remember that even though you are viewing the *subdivide* node, selecting the *polyextrude* node gives you access to its handles and parameters.

You might think that with this primitive type you are all set but it is really just a truncated icosahedron with flat faces. You need a round soccerball so you will have to put a little more work into it.



03 Press **V** in the viewport and from the **Radial Menu,** select **Shading > Smooth Shaded**. You can also use the menu in the top right of the viewport to change your shading.

This soccerball looks like a cheap plastic ball rather than a proper leathery soccerball. You are now going to branch off and add more nodes to get a better look.

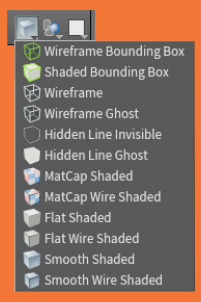
After analyzing it, set the shading back to **Smooth Wire Shaded**.

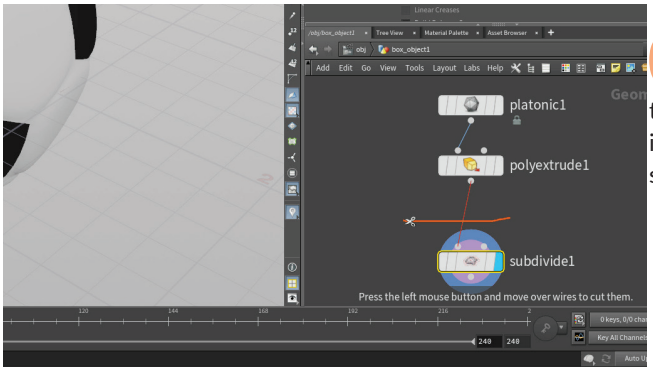
SHADING OPTIONS

There are a number of **Shading Options** available from either the **View** radial menu or the **Shading** menu in the top right of the Viewport.

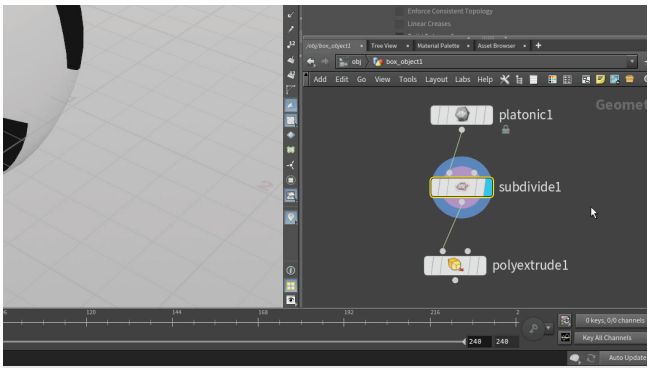
For the shading of your objects, the lighting is determined by the **Display Options** on the right edge of the Viewport. You can choose from a headlight, normal lighting or high quality lighting with shadows.

To quickly toggle from your shaded view to wireframe press the **W** key.

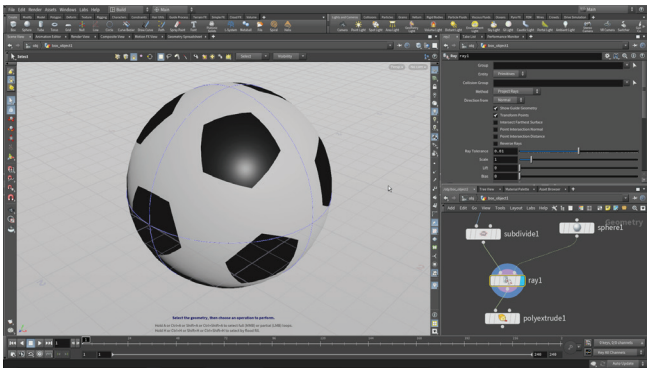




04 In the Network view, **press Y** and drag across the line connecting the *subdivide* node and the *polyextrude* node to break the connection. You are now going to move the subdivide in between the other two nodes so that you get a rounder soccerball.

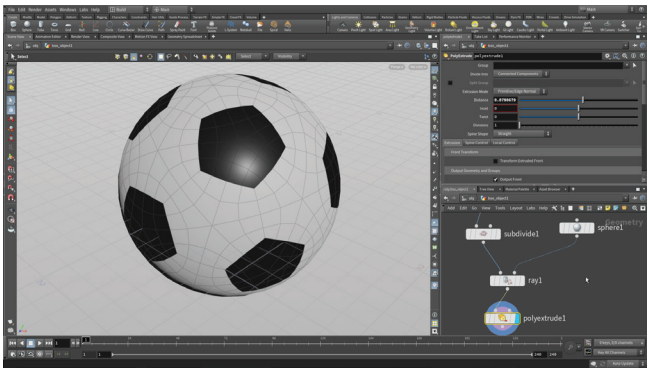


05 **Drag** the *subdivide* node in between the *platonic* solid node and the *polyextrude* node. You can drop it on the connecting wire and it will insert itself in automatically. If not then jiggle it a little until it finds the connection. This will give more detail to the sphere before it is extruded.



06 Use the **tab key** in the network editor to add a **Ray** node and wire it in after the *subdivide*. Now add a **sphere** node to the network and set its **Radius** to **1, 1, 1** and the **Primitive Type** to **Primitive**. Now wire the sphere into the second input on the *ray* node. This will project the subdivided ball onto a perfect sphere.

This is a very powerful node in Houdini that lets you project points from one piece of geometry onto another. It is the perfect solution to our problem of a subdivided soccer ball that wasn't truly round.



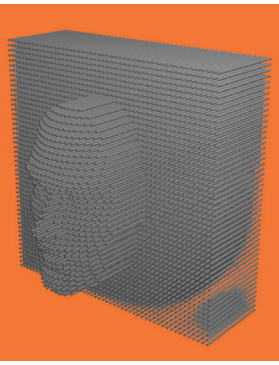
07 Set the **Display flag** back on the *polyextrude* node. With **Divide Into** set to **Individual Elements** all the small polygons are extruded but you don't want that. Set it to **Connected Components** then all the polygons are extruded.

You need a way for this network to extrude the original patches of the soccer ball but after the ball has been subdivided. You can do this using the primitive numbers on the original geometry.

THE RAY NODE

The ray node is a tool that projects points out to another piece of geometry. This is similar to the pinboard toy you played with as a kid. In fact, this is the node you would use to set up a pinboard in Houdini.

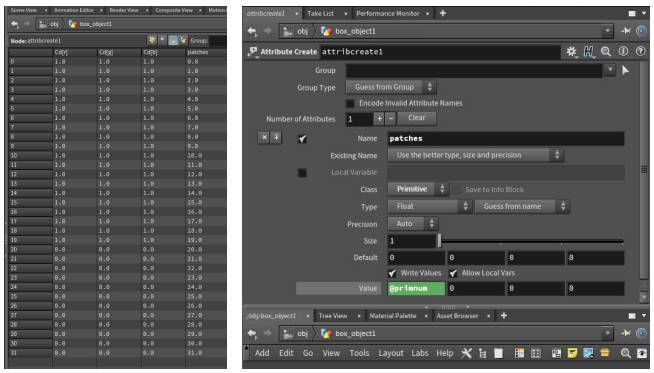
GETTING HELP | To learn more about each node, you can click on the ? help button in the top right of the Parameter pane to open up the node's online documentation. You can also hover over the tool in the shelf and **press F1**. In many cases, there are sample files that you can open in Houdini to learn what the node can do.



PART THREE

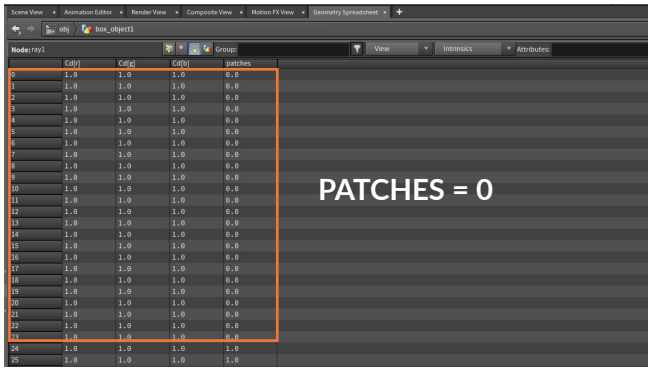
The For-Each Node

Now you get to see the magic, as the attributes you just created in the last part are fed into a for-each loop where the original patches are extruded even though each contains many polygons. This will provide a more leathery look for the soccerball once you subdivide it once more time after the poly extrudes.



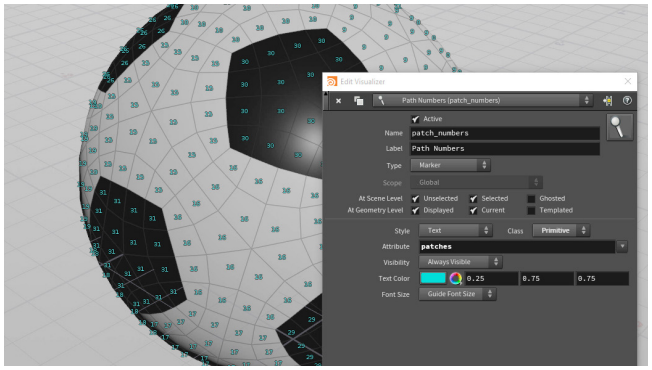
01 Add an **AttributeCreate** node after the *platonic* node. – Set its **Name** to *patches* and its **Class** to **Primitive.** Now set the first of the **Value** fields to **@primnum**.

This expression takes the primitive number attribute and turns it into a new attribute called *patches*.



02 With the *attributecreate* node selected, click on the **Geometry Spreadsheet** tab next to the main viewport. Click on the **Primitive** button and you can see the primitive numbers on the left, three color attributes which show the color of the patches and the *patches* attribute which matches the primitive numbers.

Click on the *ray* node. This attribute will be carried forward when the shape is subdivided. You can now see there are a lot more primitives but the *patches* attribute only goes as high as 31 and then it goes back to 0.

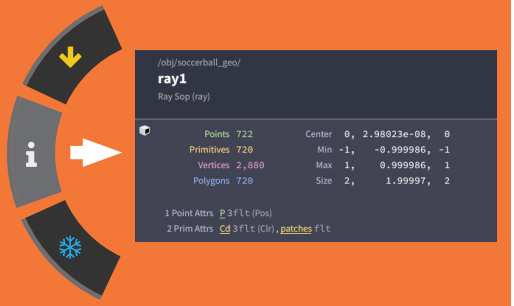


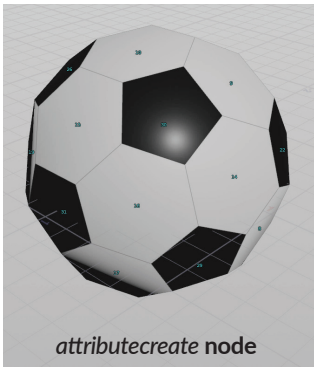
03 Go back to the **Scene View** tab, **RMB-click** on the **Visualizer display** button on the **Display Options** bar and click on the **+ Plus sign** next to **Scene** and choose **Marker.** In the Edit Visualizer panel, set **Name** and **Label** to *Patch_Numbers*, set **Type** to **Marker**, **Class** to **Primitive** and **Attribute** to *patches*.

WORKING WITH ATTRIBUTES

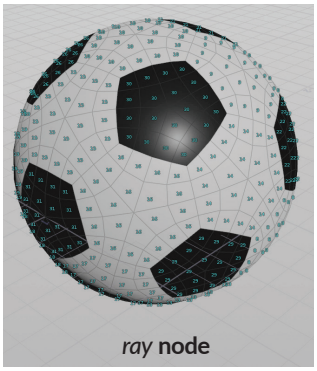
Attributes can be assigned to Points, Primitives or Vertices. Some typical types of attributes include **color (Cd)** or **UVs**. You can see the attributes at any point in your chain by mousing over a node and choosing the **i** from the radial menu. You can also review the attribute values in the **Geometry Spreadsheet** panel.

In this lesson, you will be creating a custom attribute called *patches* which will help you in the for-each loop.





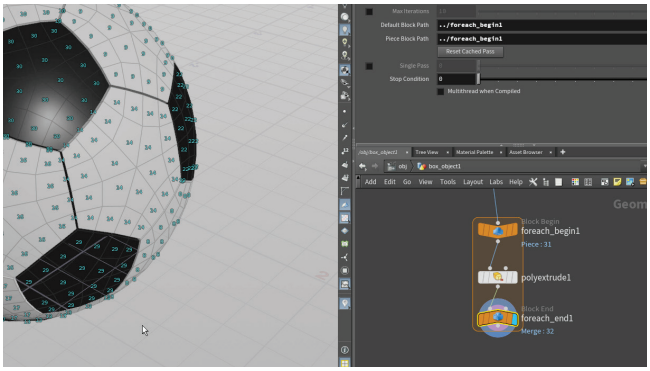
attributecreate **node**



ray **node**

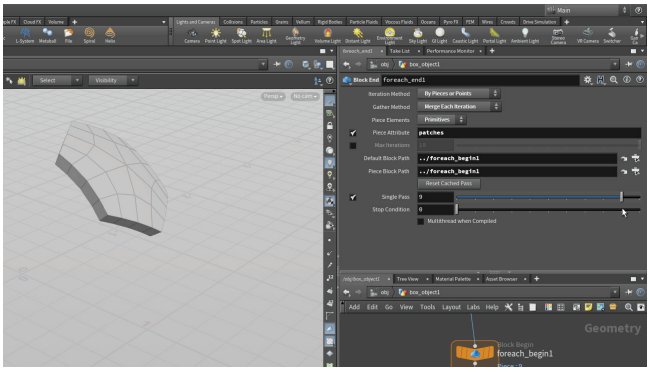
04 Now make sure the **Visualizer display** button is on and you will see the patch values on the soccerball in the viewport. You can see that the prim number from the original platonic solid have been transferred to the subdivided faces. Change the **display flag** to different nodes to see the relationship. This information will be used to polyextrude the patches properly using a for-each loop.

Turn **OFF** the Visualizer display and **save** your work. These steps have been a bit abstract and probably feel a bit too technical, but don't worry the payoff is coming.



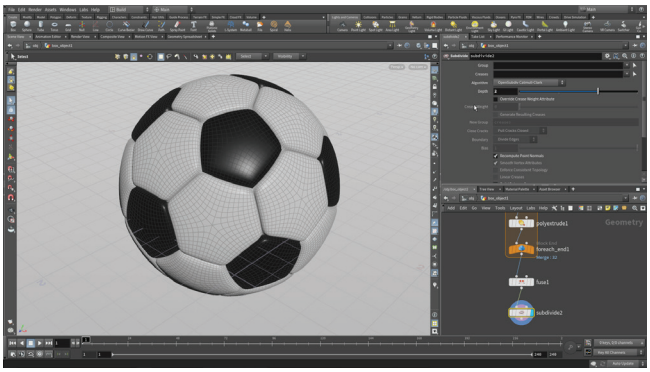
05 In the Network editor, press **tab** and start typing **For-each Named Primitive** to access two nodes that you can then place into the scene. Wire the *foreach_begin* between the *ray* node and the *polyextrude* node and then the *foreach_end* after the *polyextrude* node. Select the *foreach_end* node and in the parameter pane leave **Piece Elements** set to **Primitives** and set **Piece Attribute** to *patches*. Set display on the *foreach_end* node.

You should now see the original patches being extruded together based on the *patches* attribute. If they are not make sure **Divide to** is set to **Connected Components** on the *polyextrude* node.



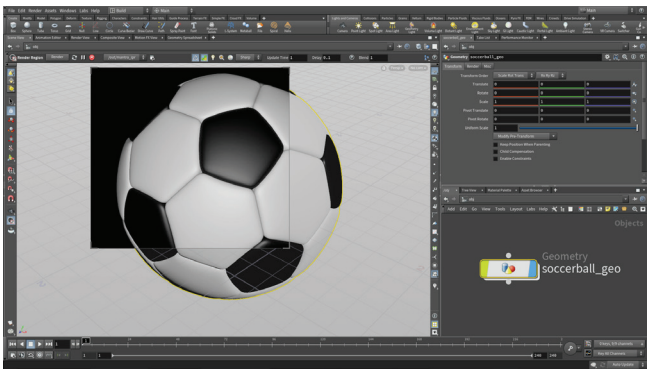
06 Click the checkbox next to **Single Pass** to explore what is happening. Drag on the slider to watch as each of the patches is polyextruded individually. You can also set values higher than 10 to see more of the patches.

Turn off **Single Pass** to see the full shape. The *for-each* nodes create all of the patches then return the final geometry. The for-each loop is a powerful set of nodes that you will use often with Houdini.



07 Add a **Fuse** node after the *foreach_end* node and set its **Display flag**. This connects the pieces into a single topology. The *for-each* nodes broke them into the different patches but didn't fuse them back together.

Add a **Subdivide** node after the Fuse. Set the **depth** to **2**. This will give you more detail in the viewport that you can use to evaluate your model. This adds more polygons but will not yet render as a true subdivision surface. Houdini also lets you set Subdivision display in the viewport without adding geometry but this Subdivide node is needed for later in the lesson.



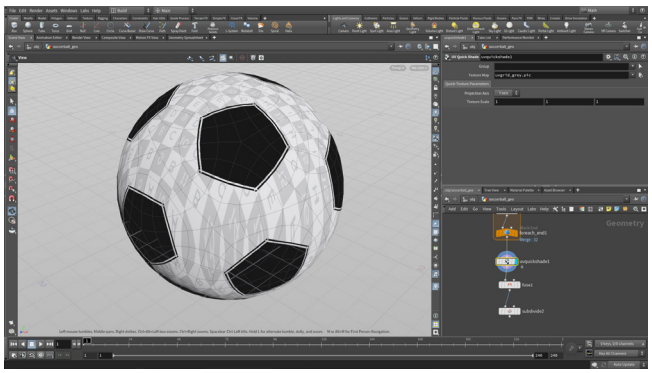
08 Play with *polyextrude* values to get a nice leathery soccer ball. Here you set a **Distance** of **0.1** and an **Inset** of **-0.02**. This gives nice rounded patches that look much better.

Go to the **Object level** and rename the object *soccerball_geo* in the Parameter pane. Either select the node and press **F2** or **double click** on the name. Click on the **Render** tab and turn on **Render Polygons as Subdivisions (Mantra)** to set up true subdivisions at render time. Select the **Render Region** tool, then draw a box around the soccer ball in the viewport create a preview rendering. To cancel, click on the **x button** in the top right of the region.

PART FOUR

Setting up UVs

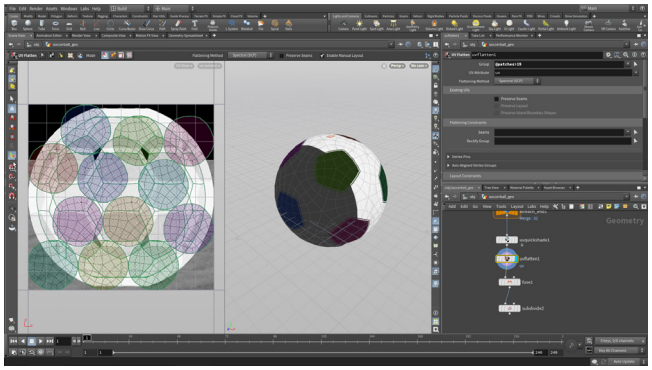
In order to set up materials and textures, it is important to make sure that there are proper UVs set up on your object. Geometry in Houdini does not come with UVs, therefore you must create them yourself. This means adding extra nodes to the network, which in this case means adding UV Quickshade and UV Flatten nodes.



01 Dive into the *soccerball_geo* object. In the Network view. press tab > **UV Quickshade** and place the new node right after the *foreach_end* node. Set its **Display Flag**.

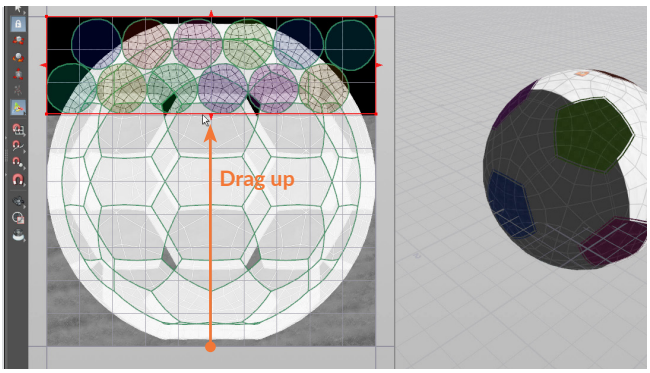
Click on the **File Selector** button next to **Texture Map**. Navigate into the `/tex` folder and select *soccerball_color.rat*. Now you can see this texture map on your geometry but everything seems stretched because the UVs were created using a projection method.

Note: *Setting up UVs to match an existing texture is not the normal order of operations. We are taking this opposite approach so that you don't need to paint a texture yourself.*



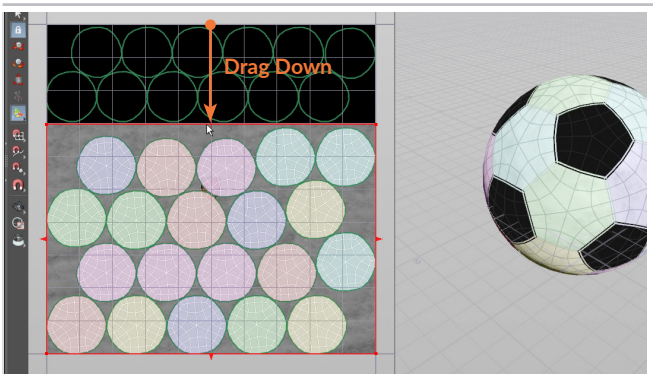
02 In the Scene view, **press n** to select all then press **tab > UV Flatten**. In the **Group** field, enter the expression `@patches>19`. This will flatten the dark patches on the soccerball geometry using the patch boundaries to lay out the UVs.

Using this tool brings up UV view. Click on the **UV (vertex)** menu in the top right and choose **Background > *soccerball_color.rat***. Now you can see this texture in the background of the UV panel. This texture has a team insignia at the center of the image and a darker area at the top for the dark patches.



03 **RMB-click** on the Handle tool and turn on the *Min*: handle. Use the arrow handle at the bottom and pull it up until all the patches are inside the dark area at the top of the texture map.

If you had placed this node after the *fuse* or *subdivide* then there wouldn't have been boundaries to work with. Houdini's ability to let you set up UVs in the middle of an existing network offers a lot of flexibility.

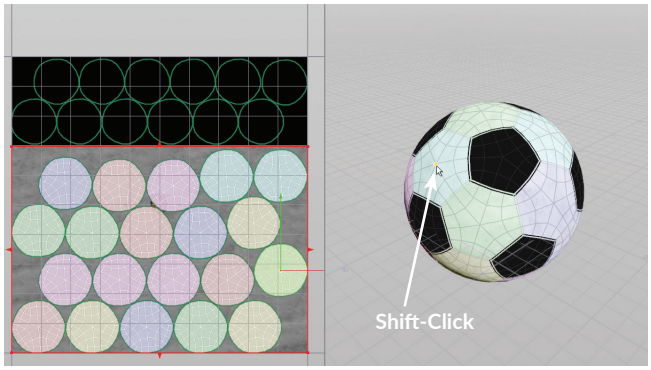


04 Add another **UV Flatten** node after the first *uvflatten* node and before the *Fuse* node. In the Group field, enter the expression `@patches<20`. This will flatten the light patches.

RMB-click on the Handle tool and turn on the *Min*: handle. Use the arrow handle at the top and pull it down until all the patches are inside the light area at the bottom of the texture map.

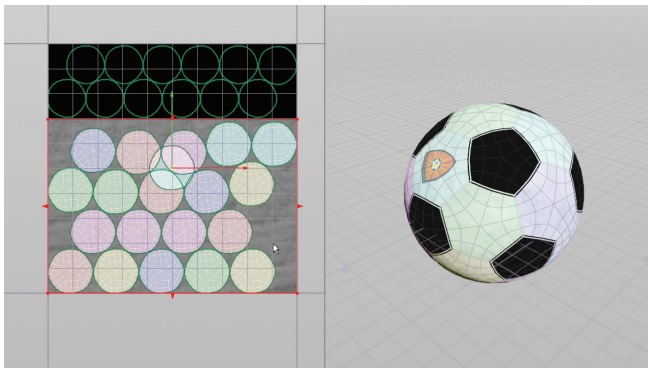
When you finish you can **RMB-click** on the Handle tool and turn off the *Min*: handle.

RMB-click in the Scene view and from the menu, select **Texture Visualization > Off**. This will remove the grid.



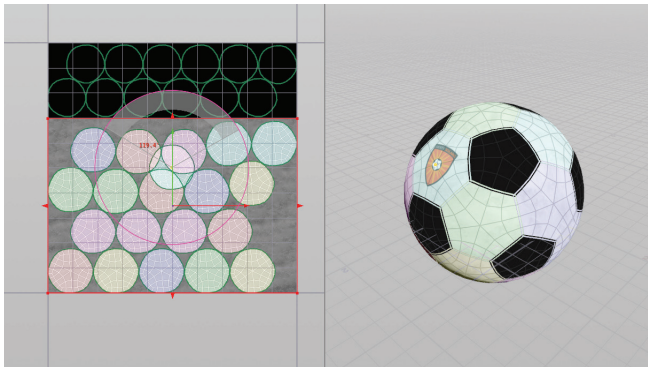
05 Select the *uvflatten* node and click on the **Handle** tool. Mouse over the geometry you can see the patches highlight. Click on the **Pin Vertices** button in the **Operation Controls** bar at the top of the viewport.

Mouse over the patch shown in this image and in the 3D view, **Shift-click** on the center vertex of the desired patch to select it. This adds a pin in the UV view and a handle to work with.

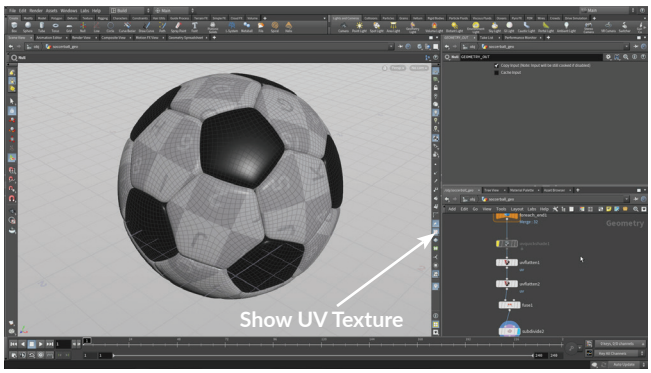


06 Go to the UV view and **move** the pin and the patch so that it is centered on the logo. **Press Y** to get a rotate handle and rotate the patch until you line up the logo.

By pinning vertices on patches in the UV view, you can lock down their position. At first there is some overlap of the neighboring patches but you can easily fix this by repacking.



07 Click on the **Repack** button in the **Operation Controls** bar to reorganize the other patches around the new one. You can continue moving the patch around using the pin but will need another **Repack** to avoid overlapping UVs.



08 Set the *quickshade* node to **Bypass** to hide the assignment of the texture map. This node was only needed when you were setting up the UVs. You now see a UV grid on the soccerball.

To hide the UV display in the perspective view, got to the **Display Options** bar and turn **Off** the **Show UV Texture when UV's Present** button.

Add a **Null** node to the end of the chain and call it *GEOMETRY_OUT*. It is a good idea to have this kind of node to define the end of the network chain. **Save** your work.

UV FLATTEN

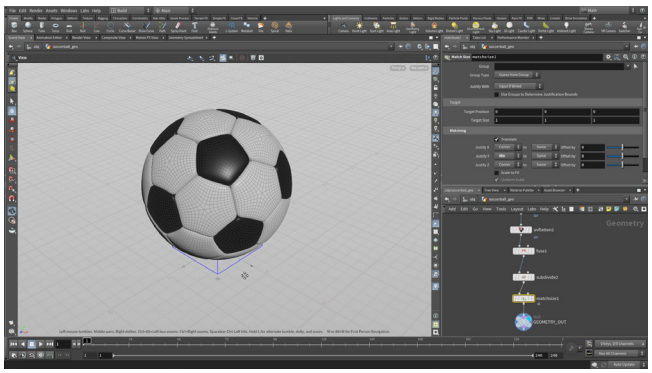
The **UV Flatten** node works in two steps. It takes individual texture pieces, defined by seams, and flattens them into 2D texture space, trying to equalize polygon size.

This node lets you add constraints for the flattening algorithm. Constraints force the layout algorithm to satisfy certain conditions, giving you extra control over the final UV layout. You can use this node interactively, using the tools in the node's state to specify constraints, or you can turn off Manual layout and use the node procedurally.

PART FIVE

Layout: Cameras and Lights

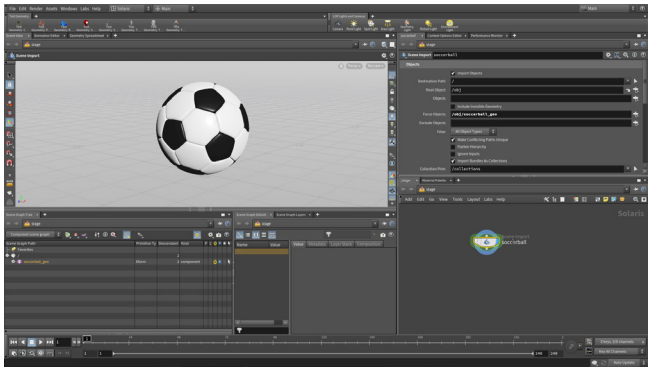
To create a scene for rendering, you are going to bring the geometry into the Solaris or LOPS context of Houdini. This is an environment dedicated to lookdev, layout and lighting and is built on the foundation of USD (Universal Scene Description). This will allow you to render to the Karma renderer which works right in the Scene View as part of the Solaris workflow.



01

In the Network View, press **tab > Match Size** then add the node between the *subdivide* and *GEOMETRY_OUT* null. Set the **Display flag** on the *GEOMETRY_OUT* node.

Select the *matchsize* node and set **Justify Y** to **Min** to raise the ball up to sit on the ground. This will put it in the right position for rendering.



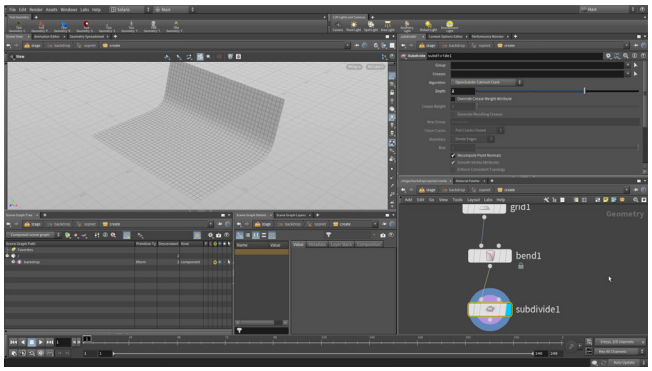
02

Change the desktop to **Solaris**. Make sure you are looking at the **Stage** in the path bar.

In the Network view, press **tab > Scene Import** and click to place the node down. Next to the **Force Objects** field, click on the **node selector** button and from the pop up window, select the *soccerball_geo* object then click **Accept Pattern**.

In the Scene View, use your view tools such as **spacebar-h** for homing the view to get a better look at the soccerball.

Note: *Force Objects is used instead of the Objects field because it will bring the object into LOPS even if the object's display flag is off.*

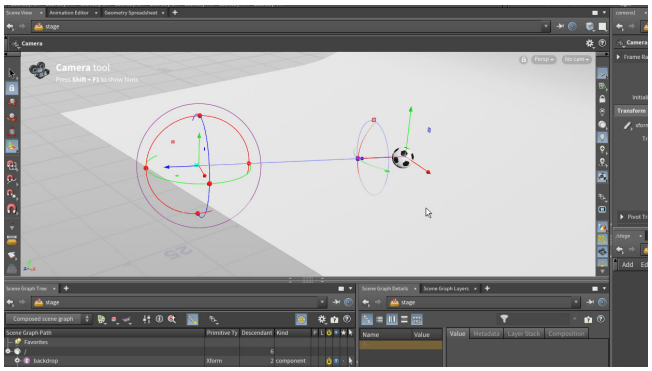


03

In the Network view, **press tab** and type out **Grid**. Click to place down the node and rename it *backdrop*. Double-click on the *backdrop* node to dive down to the geometry level.

Select the *grid* node and set the size to **80, 80** and the **Center** to **0, 0, -20**. **RMB-click** on the *grid* node's output and type **Bend**. Click to place the node then set its **Display Flag** and set: **Bend** to **75.** In the **Capture** section, set **Capture Origin** to **0, 0, -30, Capture Direction** to **0, 0, -1,** and **Capture Length** to **5.**

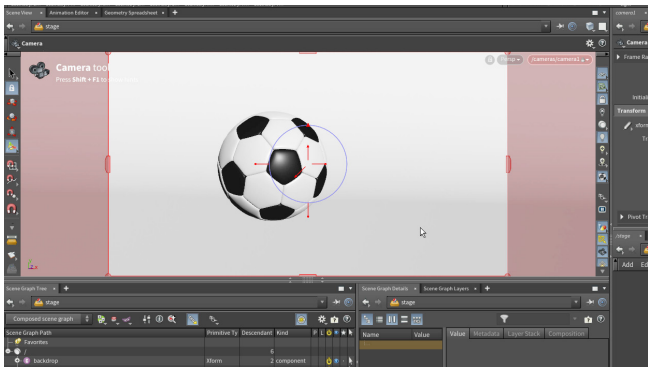
RMB-click on the *grid* node's output and type **Subdivide.** Set its **Display Flag** then set **Depth** to **2.**



04

Go back to the **Stage** level. Wire the *backdrop* node into the *sceneimport* node. **RMB-click** on the output of sceneimport and type out **Camera**. Press **Enter** to place the node then set its **Display Flag**.

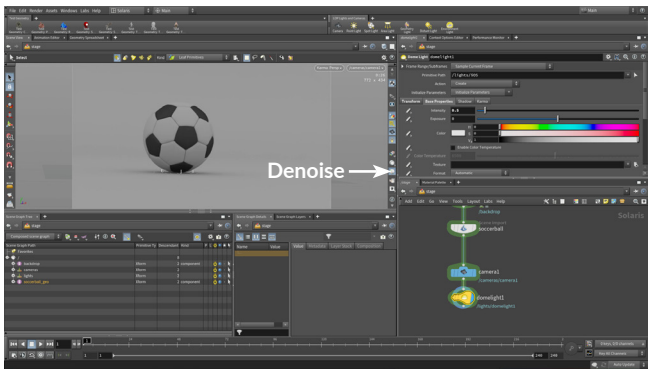
In the Scene View, you will see camera handles at the origin. Zoom out and look down at the scene then adjust the handles so the camera is looking at the soccerball from the left. You may want to activate the **Construction plane** so that you move the handles along the ground. You can also use the axis handles to control the direction and lift the camera up from the ground.



05 In the top right of the **Scene View**, click on the *No cam* menu and choose *camera1*. Now you are looking through the camera and can adjust how it looks.

This is probably not the view you are looking for therefore some view changes are needed. On the right side of the Scene View, click on the **Lock camera to view** button. Now use the **View tools [Spacebar-LMB/MMB/RMB]** to reposition the camera.

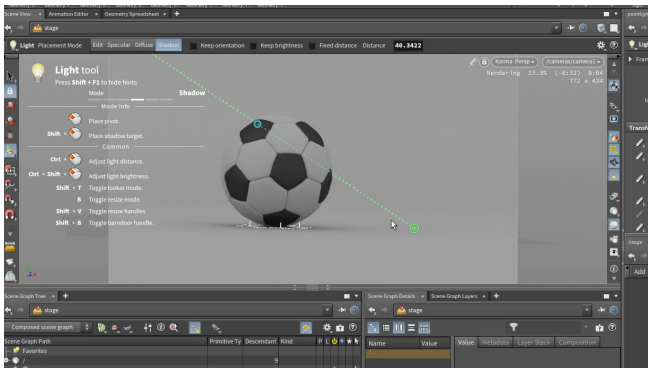
IMPORTANT: When you finish, toggle off the **Lock Camera** button.



06 From the **LOP Lights and Camera** shelf, **click** on the **Environment Light** tool then press **Enter** to place it at the origin. Set the **Intensity** to **0.5** to tone it down a bit.

Now click on the menu just to the left of the camera menu, and set it to **Karma**. Now you are using the Karma renderer in the viewport.

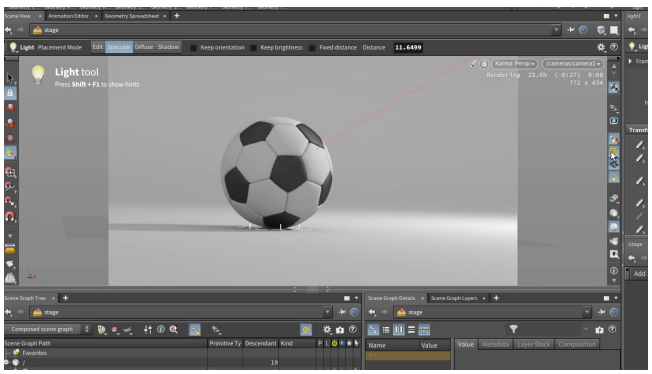
If you have an Nvidia graphics card and have the latest drivers installed, you can turn on the Optix Denoiser to resolve the image more quickly. Turn it on in the **Display Options** bar or **press d** and setting **Enable Denoising** in the **Render Display Options**.



07 From the **LOP Lights and Camera** shelf, **click** on the **Point Light** tool then press **Enter** to place it at the origin. You are looking through the light. Set the camera back to *camera1*.

With the node active, press **Shift-F** to turn on the **Shadow** mode. You can also click on it in the **Operation Control** bar. Now **click** on the top of the soccerball to set a pivot point then **Shift-click** to place a target on the ground. **Ctrl-Drag** to set the light distance.

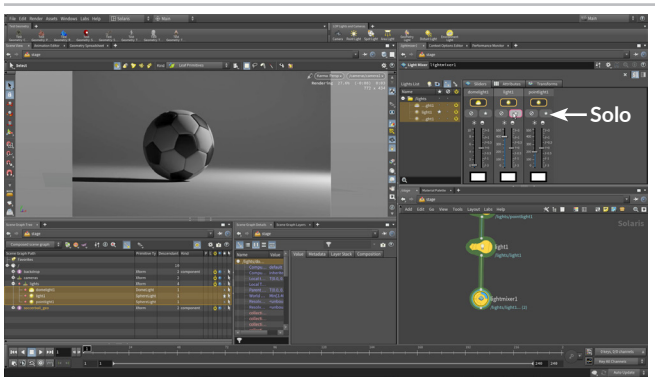
Now you can use **Ctrl-Shift-drag** to change the intensity of the light. You may need to set it quite high go see some impact on the look of the soccerball.



08 In the Network view, **RMB-click** on the *pointlights* node's output and type **Light** then press **Enter** to place the node and set its **Display Flag**.

With the node active, press **Shift-S** to turn on the **Specular** mode. Now click on the right side of the soccerball to define a specular area of focus.

Now you can use **Ctrl-drag** to move the light away from the soccer ball and **Ctrl-Shift-drag** to change the intensity of the light.



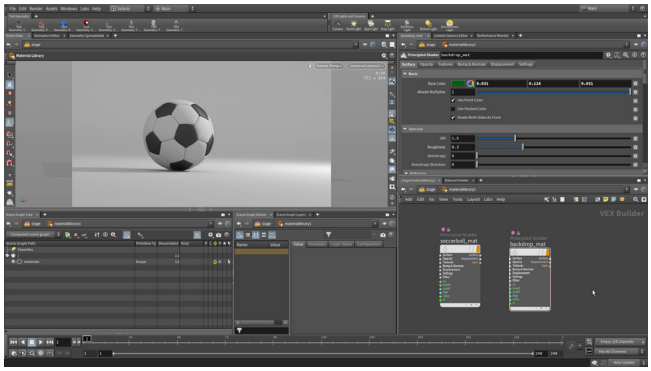
09 In the Network view, **RMB-click** on the *lights* node's output and type **Light Mixer** then press **Enter** to place the node and set its **Display Flag**. This will create a special panel in the Parameter pane which has a list of lights on the left side.

Drag the three lights from the list to the area on the right. Click on the **Star** icon to **Solo** each light to determine its contribution then tweak **Exposure** to adjust the lighting. Since the **intensities** are so high you can click on the icon above the intensity bar and from the pop-up set a **Max value** that works for your shot. When you are finished be sure to **turn off** the **Solo** button to see all the lights.

PART SIX

Lookdev: Materials

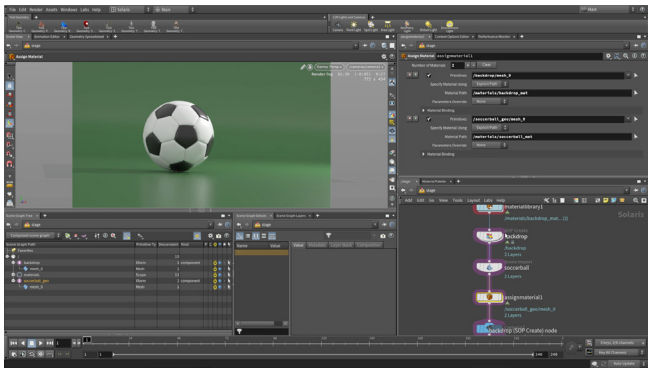
Materials and shaders can also be created within the LOPS/Solaris context. This involves adding the materials to the Scene Graph then assigning them to the geometry. The Materials are created inside a Material Library node then assigned at the LOPS/Solaris level. To add textures to the backdrop, UVs will have to be created to position the maps properly.



01 In the Network view, press **tab > Material Library** and place the node just above existing network of nodes then wire it into the *backdrop* node. **Double click** on the node to go down to the **VEX Builder** level.

Press **tab > Principled Shader** and place the node down. Rename it *soccerball_mat*. Change its **Base Color** to **white (1, 1, 1).**

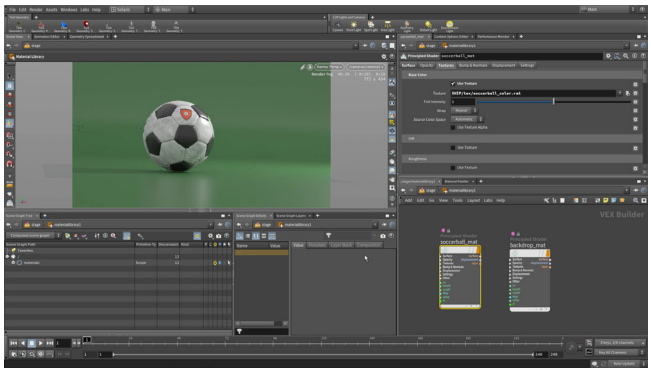
Alt-drag on this node to create a second principled Shader and rename this one *backdrop_mat*. Change its **Base Color** to a dark green.



02 Go back to the **Stage** level then press **tab > Assign Material** and place the node in between the *sceneimport* and the *camera* nodes.

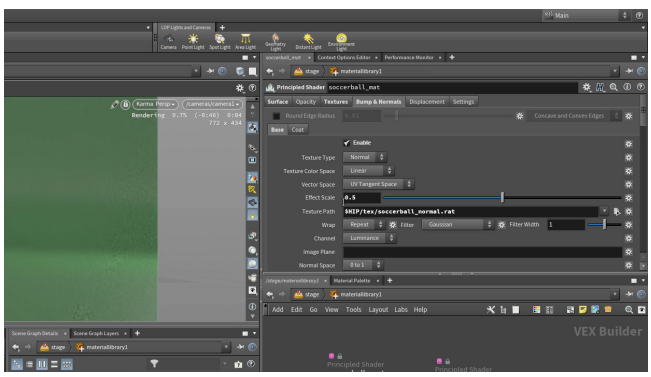
In the Parameter pane, next to **Primitives**, click on the **arrow** button and in the viewport select the *soccerball* geometry. **Press Enter** to add its path to the **Primitives** field. Now click on the **arrow** next to **Material Path** and from the pop-up window, select *materials* > *soccerball_mat* and click **OK**.

Click on the **+ button** then repeat these steps for the *backdrop* and the *backdrop_mat*.



03 Double-click on the *materiallibrary* node to dive into it and select the *soccerball_mat* node. Click on the **Textures** tab and under **Base Color** click on **Use Texture** then use the button next to **Texture** to call up the file window. Click on **$HIP** in the side list then click on the *tex* folder to open it and then click once on *soccerball_color.rat* to select it. Click **Accept** to assign the texture to the material.

The $HIP reference makes sure that the reference is relative to the location of your scene file. That way if you were to move your project directories to another computer the reference will still work.



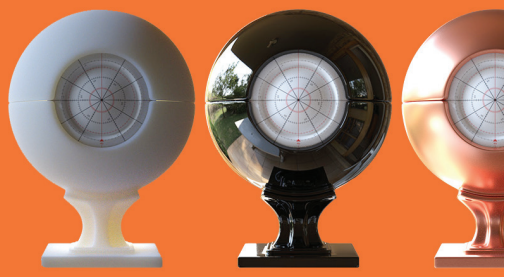
04 Under the **Textures** tab and use the technique you learned in the last step to assign textures to **Roughness** and **Reflectivity**. You will find the appropriate textures in the *tex* folder.

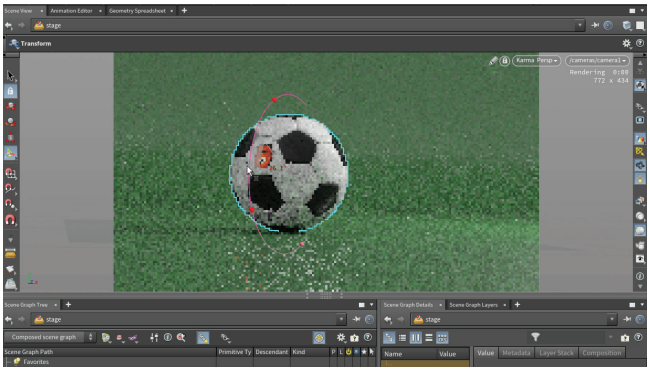
Go to the **Bumps & Normals** tab on the material and click the **Enable** button. Click on the arrow next to **Texture Path** and from the tex directory choose the *soccerball_normal.rat* file. Set the **Effect Scale** to around **0.5** and see how it looks.

MATERIALS IN HOUDINI

Materials in Houdini live in the **VEX Builder** context which in this case is nested inside the **Material Library** node. A material is made up of **VOP** nodes or **Material X** nodes that define the material qualities.

The **Principled Shader** is an uber material that can be used on its own to assign texture maps and achieve a large variety of looks. You can also build your own shaders and materials for more advanced looks.



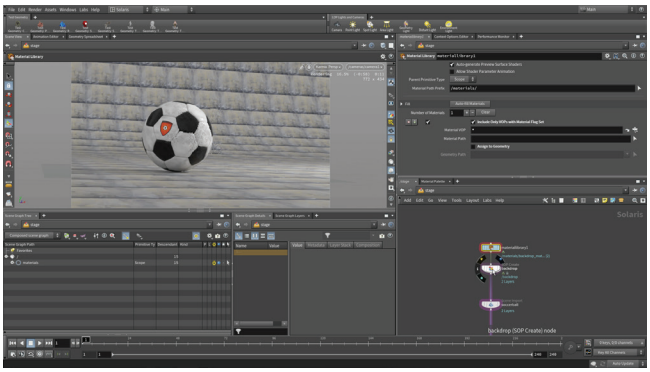


05

In the Scene view, look through *camera1* to see your shot. The position of the soccerball's logo isn't quite right in relation to this camera therefore you need to rotate the ball.

In the Scene view, select the soccerball. **Press tab > transform**. This adds a transform node to the end of the chain. With the **Handle** tool, **press r** to get the **Rotate** handle and rotate the ball see the logo properly. You may need to tumble around to get this to work.

You can do this in the **Karma** or **Houdini GL** view. If you don't like the edits then you can delete the node and try again. To deselect the *soccerball_geo*, press **Ctrl** and click on it in the **Scene Graph**.

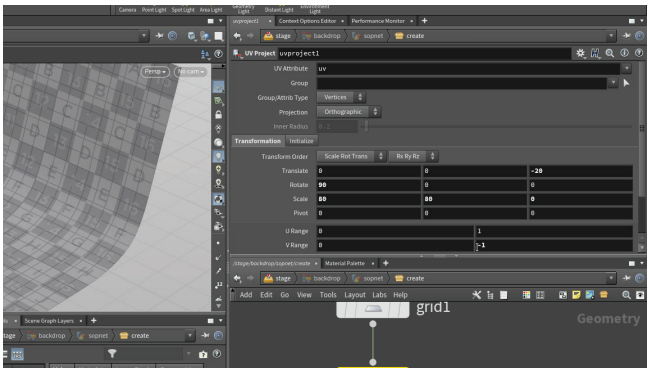


06

Now lets add some texture maps to the Backdrop material. **Double-click** on the *materiallibrary* node and select the *backdrop_mat* node.

Set the **Base Color** to **1, 1, 1** because this color will be multiplied with the texture map. Now click on the **Textures** tab and under **Base Color** click **Use Texture**. Click the **File Selector** button and use `$HIP` to go to the `/tex` directory and choose *backdrop_color.rat*. You can also add the *backdrop_reflect.rat* texture to **Reflectivity**.

You can see in the **Scene View** that UVs are not set up properly and the texture map isn't working.

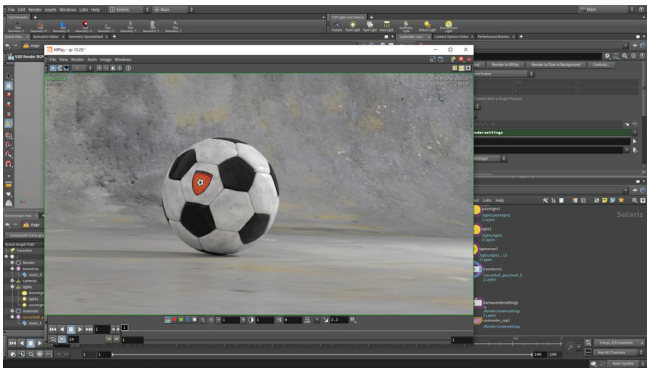


07

Go back to the Stage level then **double-click** on the *backdrop* node to go to the geometry level. Add a **UV Project** node between the *grid* and the *bend*. You are putting the nodes here so the UVs are created before the surface is bent.

On the *uvproject* node, click on the **Initialize** tab and click the **Initialize** button. Go back to the **Transformation** tab and set **V Range** to **0, -1**. This will orient the UVs properly.

IMPORTANT: Set the **Display Flag** back to the *subdivide* node.



08

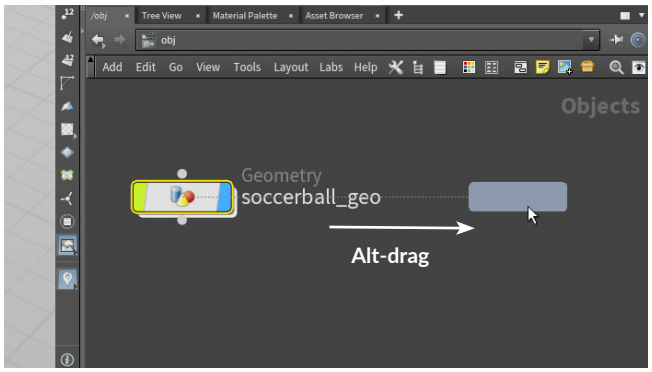
Go back to the Stage level. In the **Network View**, press **tab > Karma** to add a **Karma Render Settings** and **USD Render ROP** node. Wire them into the end of the chain. Select the *karmarendersettings* node and on the **Image Output > Filters** tab set **Denoiser** to **nvidia Optix Denoiser** to turn the denoiser back on.

Select the *usdrender_rop* node. Click on the **Render to Mplay** button. This opens **Mplay** which shows the rendering as it progresses. Choose **File > Save Frame As** to save the image to disk.

PART SEVEN

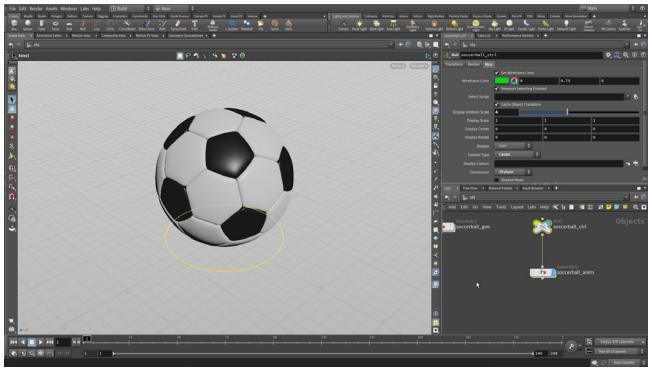
Rig the Soccerball

In order to create an animation of the ball bouncing, you will start by building a simple rig that will make it easier to keyframe. This will involve setting up null objects so that you can work interactively in the viewport and adding nodes to the soccerball geometry network to accommodate the ball rotation along with squash and stretch.



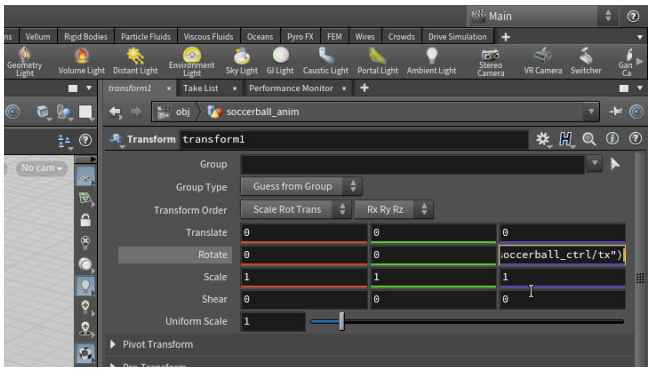
01 Change back to the **Build** desktop and navigate to the object level by clicking on one of the path bars and choosing *obj*. Now in the network editor, **Alt-drag** on *soccerball_geo* to make a copy of it. Rename this node socccerball_anim.

You will use *soccer_anim* to build your rig. Now turn off the Display Flag on the *soccer_geo* node to hide it. You don't want to make changes to the original setup because that object is being used in SHOT 1 in the Solaris context. This new soccerball will be used for an animated SHOT 2.



02 On the **Create** shelf, click on the **Null** tool then press **Enter** to place it at the origin. Name it *soccerball_ctrl*. Go to the **Misc** tab and set **Control Type** to **Circles** and **Orientation** to **ZX Plane**. Set the **Display Uniform Scale** to **4**. This creates a handle for the rig that is easy to select that won't render later on.

In the network editor, connect the input of the *soccerball_anim* object to the output of the *soccerball_ctrl* null to create a child/parent relationship. Moving the null will move the ball. Turn off the **selection flag** on the *soccerball_anim* so that you don't select it by accident in the viewport while animating. You will use *soccerball_ctrl* instead.

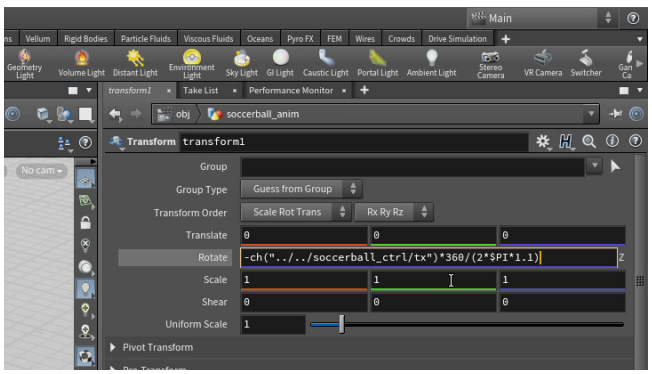


03 Select the *soccerball_ctrl* node. In the Parameter pane, click on the **Transform** tab then **RMB-click** on the **Translate X** parameter. Choose **Copy Parameter**.

Dive into the *soccerball_anim* object. Add a **Transform** node between the *subdivide* and the *matchsize* nodes. **RMB-click** on the **Rotate Z** and choose **Paste Relative References**. This places a channel reference expression in this parameter.

```
ch("../../soccerball_ctrl/tx")
```

This will connect the movement of the control object to the rotation on this node.

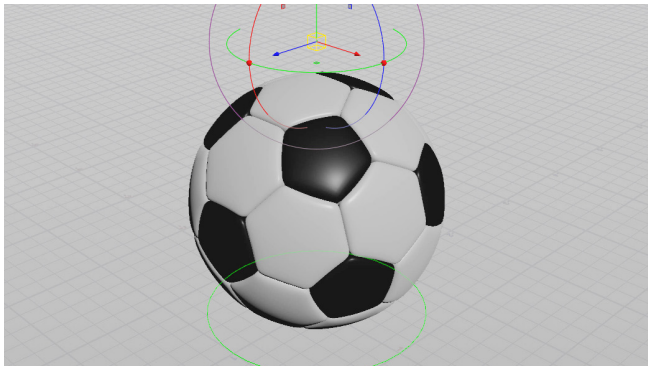


04 Click on the parameter to expand the channel. You are now going to use the **ball's circumference (2πr)** to determine the ball's rotation as it moves forward.

Edit the expression to read:

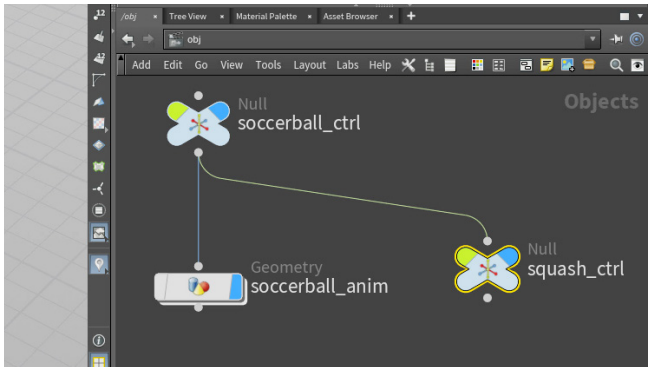
```
-ch("../../soccerball_ctrl/tx")*360/(2*$PI*1.1)
```

First you add a negative **(-)** at the front. You then multiply the position of the ball by **360** degrees and divide by **2πr** - π being **$PI** in the expression. At the object level, move the *soccerball_ctrl* along the **X axis**. The expression will rotate the ball to match the motion. Put it back at the origin when you are finished.



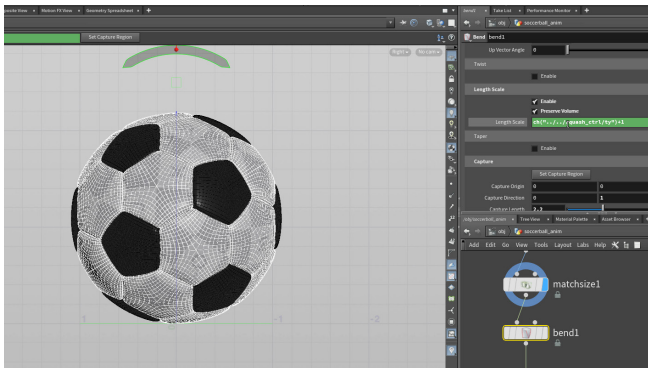
05 In the Viewport, create another **Null** object at the origin. Call it *squash_ctrl*. Go to the **Misc** tab and set **Control Type** to **Box** and the **Display Uniform Scale** to **0.2**.

Move the null up just above the ball. **Translate Y** should be about **2.5**. In the Parameter pane, choose the **Modify Pre-Transforms** menu and select **Clean Translates**. This sets the **Translate Y** value of the null to **0** even though it is above the ground. In order for this null to drive the squash and stretch, it needs a default value of 0.



06 Parent the *squash_ctrl* null to the *soccerball_ctrl* null. This will ensure that this secondary null moves when you animate the control null.

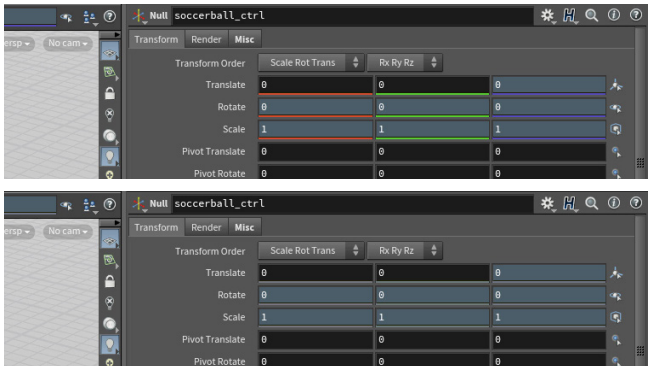
RMB-click on the *squash_ctrl* node's **Translate Y** parameter. Choose **Copy Parameter**. You will use this parameter to drive the squash and stretch of the ball. This will allow you to control the squash and stretch from the viewport interactively.



07 Go into the *soccerball_anim* object. Add a **Bend** node after *matchsize*. Turn **Off** the **Limit Deformation to Capture Region** checkbox.

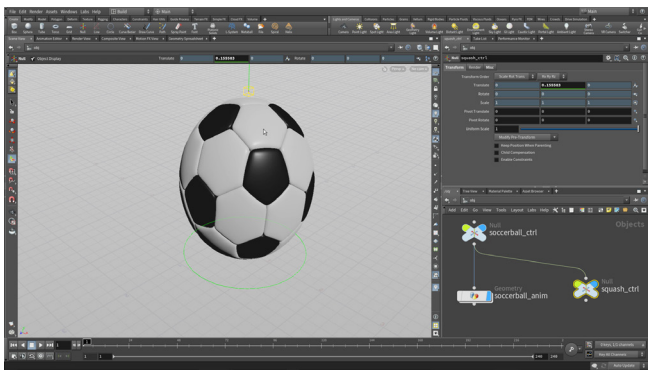
Go to a **Right View** then click the **Set Capture Region** button. Turn on **Grid Snapping** and place a point at the base of the ball and another at the top. This should set **Up Vector** to **0, 0, 1**, **Capture Direction** to **0, 1, 0** and **Capture Length** to **2.2**.

Turn on **Length Scale** and **Preserve Volume** then **RMB-click** on **Length Scale** and choose **Paste Relative References**. Add a **+1** at the end of the expression.



08 Go to the Object level and a perspective view. **RMB-click** on the **Transform X** and **Transform Z** parameters and choose **Lock Parameter** to lock these parameters on *squash_ctrl*. **RMB-click** on the **Scale** and **Rotate** parameters and choose **Lock Parameter** to lock all three channels.

Select the *soccerball_ctrl* object. **Lock** all the channels except **Translate X** and **Translate Y.** Now when you select the controls you will only see handles for the unlocked channels. This will make it easier to work with the rig because the animator can only manipulate the chosen parameters.



09 Now test out the rig by moving it around in X and Y and using the second handle to squash and stretch it. Once you are sure that all the parts are working, return all the values to 0 and get ready to animate.

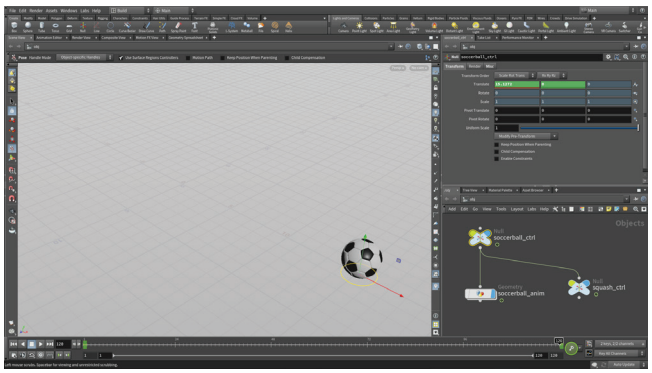
You may want to turn off **Secure Selection** in the toolbar on the left side of the Scene view. This will make it easier to select the two control nulls while in the **Move** tool. If not then you will need to press the **S key** every time you want to switch selections.

Save your scene file before proceeding.

PART EIGHT

Animate a Bouncing Ball

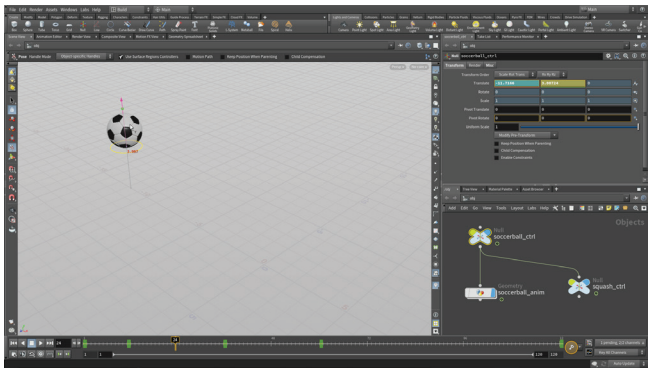
You can now take the soccer ball rig and use it to animate the ball bouncing. You will learn how to set keyframes, adjust animation curves and work with time-space handles in the viewport. The bouncing ball is a classic animation exercise that offers a great opportunity to learning the basics of animating in Houdini.



01 At the bottom left edge of the **Timeline**, click on the **Global Animation Options** button. Set the *End* to **120** and click **Close**. This will set the timeline range to 120 frames.

Make sure you are on **frame 1**. Click on the **Pose** tool on the toolbar to your left and select the *soccerball_ctrl*. Move the ball to around **-15** in X and **press K** to keyframe it. Move the timeline to **frame 120.** Move the ball to the around **15** in X and **press K** to set a second keyframe.

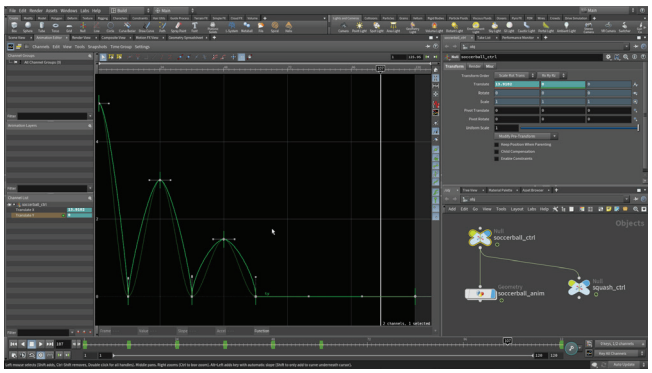
Scrub through the timeline to make sure that the ball is animating. It should be moving and rotating based on the rig design.



02 Move the timeline to frame **12** and **press K** to set an intermediate key. Repeat at frames **36** and **60**. All of these keyframes are sitting on the ground.

Now go to **frame 1** and lift the ball up in the Y direction. You don't need to set another key because this move simply updates the keyframe you already set at frame 1 .

Move to **frame 24** and lift up the ball in the Y direction a little less than you did at frame 1. **Press K** to set a keyframe. Move to **frame 48** and lift the ball up even less. **Press K** to set another keyframe.



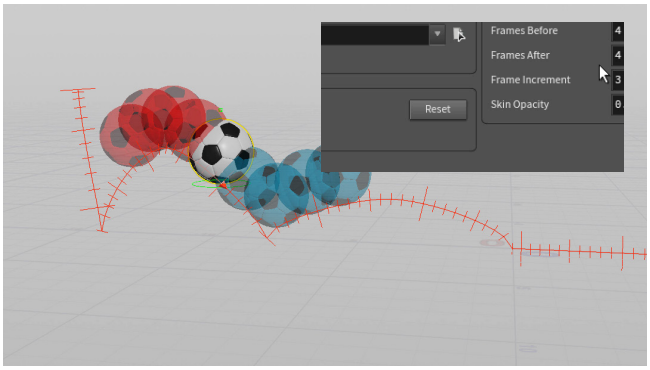
03 **Scrub** through the timeline to see that the ball appears to float whereas you want hard hits when the ball contacts the ground. Click on the **Animation Editor** pane tab.

From the Scoped parameter list, click on the *Translate Y* channel. **Press H** to home the view of the curve. Select the three keyframes where the ball contacts the ground and press the **Untie Handles** button on the **Functions** bar found just above the graph. Now click in empty space to deselect then start tweaking the tangent handles to create a sharp bounce at each point. You can also stretch out the handles at the top to slow the ball down at the peak.

THE POSE TOOL

You can easily manipulate the two control objects in the soccerball rig using the **Move** or **Handle** tool. The advantage of using the **Pose** tool is that it gives you access to the **Motion Path** handle and if you were working with Inverse kinematics, there are special handles that you can use to control the system. Therefore be sure to remember this tool when you start setting keyframes on your rigs. The **Secure Selection** does not restrict the **Pose** tool from selecting different objects.

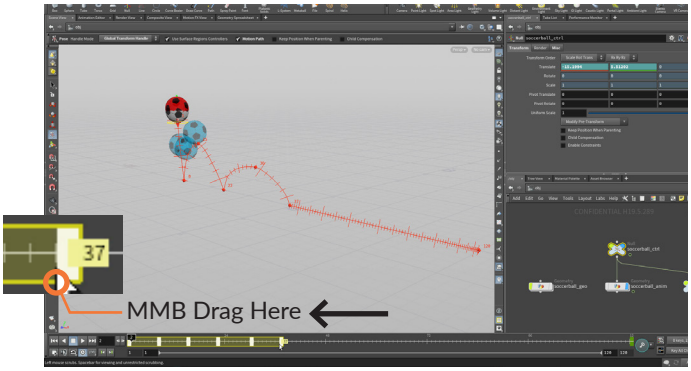
Select
Secure Selection [~]
Move [T]
Rotate [R]
Scale [E]
POSE [Y or Ctrl R]
Handle [Enter]



04

Go back to the **Scene view** and preview the results. You now have a sharper hit each time the ball hits the ground. Make sure you have the **Pose** tool active and the *soccerball_ctrl* node selected. In the top bar, turn **On** the **Motion Path** handle. This shows a path outlined where the ball is bouncing. Click on each keyframe marker to tweak the bounce. **RMB-click** on the handle to **Show tangents** for more control over the curve.

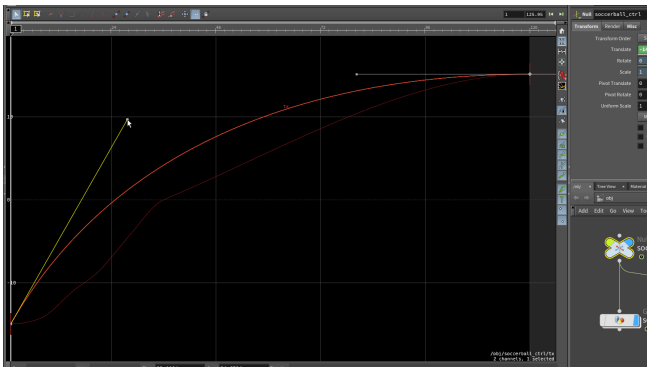
Go to the *soccerball_anim* object's **Misc** tab, set **Onion Skinning** to **Full Deformation**. Press **spacebar-d** and from the **Scene** tab adjust **Frame increment** and the **Frames Before** and **After** color.



05

To adjust the timing, you can also use the timeline. Press **Shift** and drag a bounding box from frame 1 to the last key in the timeline **to select** all the keys. Next, **MMB drag** on the end of the box underneath the handle to scale the timing of the bounces to speed them up. You can also select each key using **MMB** and then drag with **MMB** to time each keyframe the way you want.

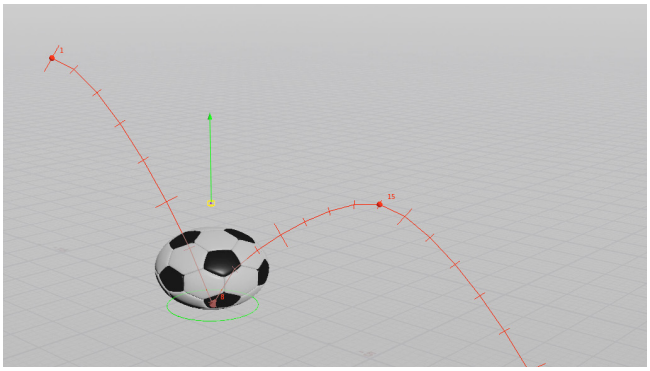
This is where you will determine the timing of the bounces. Keep exploring until you get the look that you want. Note that there may be some awkwardness in the bouncing because of the translate X values. You will fix that in the next step.



06

Click on the **Animation Editor** pane tab and you will see two curves. From the Scoped Parameters list, click on **Translate X** for *soccerball_ctrl*. Now select all the keys except for the first and the last. Press **Delete**. Now use the curve handles to go from a high slope to a low slope. This will have the ball moving faster at the beginning and slower at the end.

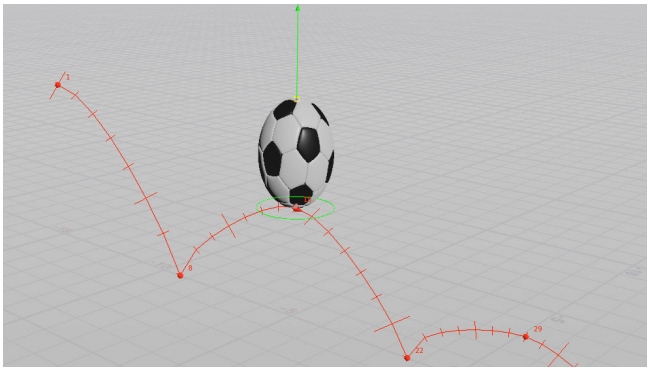
Note that if you go back to tweak the points in X on the motion path handle, you will get strange results because there are no longer intermediate keys in that direction - only use it to tweak in Y from this point on.



07

Go back to the Scene view. **RMB-click** on the **Motion Path** handle and choose **Persistent**. This will keep it around as a guide as you set keyframes on the squash and stretch.

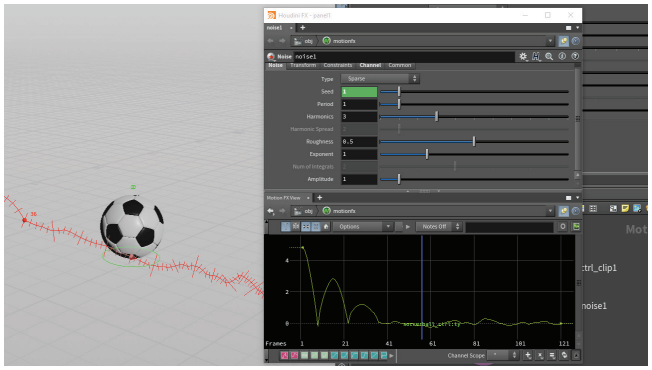
Select the *squash_ctrl* and turn off its **Motion Path**. Go to the first bounce and move back one frame. Select the *squash_ctrl* handle and stretch out the ball a bit. **Set a Keyframe** using the **K** key. Now go to the bounce frame and move the handle down to create squash. **Set another key**. Go one frame forward and stretch the ball until it is round. **Set another key**. Repeat for all the bounces.



08

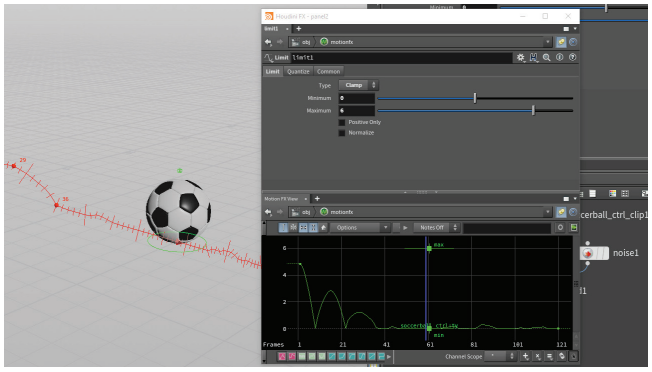
When you are finished, scrub and playback the motion to preview the results. Make sure the peaks of the bounces are stretched out. Make sure the **Real Time Toggle** is **On** in the Timeline to properly evaluate the motion.

You can now use the **Animation editor** to make tweaks to the squash and stretch. Make sure you keep the keyframes aligned with the bouncing of the soccer balls.



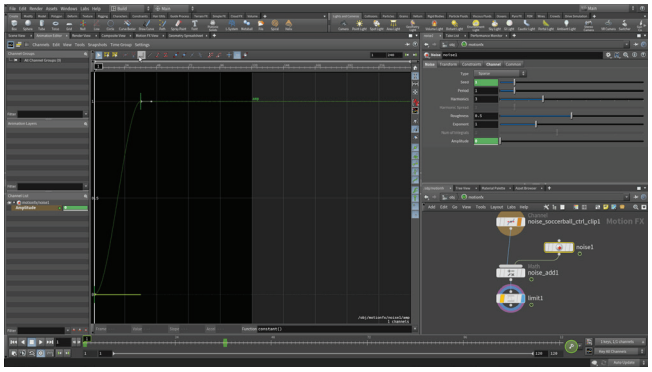
09 Select the *soccerball_ctrl* null object. **RMB-click** on the **Translate Y** and choose **Motion FX > Noise**. A panel pops up with parameters that you can use to control the noise. A new subnetwork was created with the CHOP nodes that send their information back to the *soccerball_ctrl's* **Translate Y** channel.

Set the **Amplitude** to **5** and press **Play** to see how this looks. This adds dramatic up and down motion which make it feel like there is some serious turbulence. Set the **Amplitude** to **1.** This offers a more subtle bump to the motion of the ball.



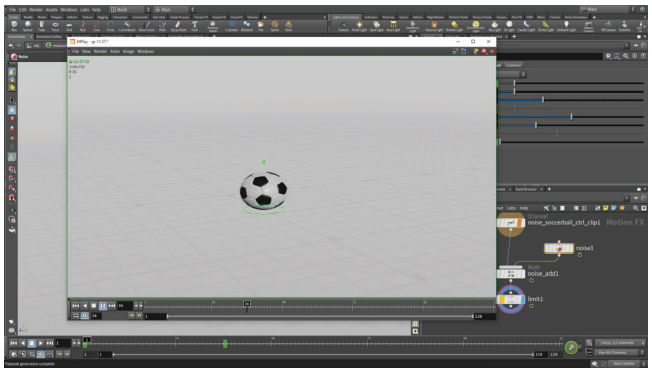
10 Some of this motion goes below the ground. You will want to focus on creating bumps above the ground.

Go back to the object level. Select the *soccerball_ctrl* null object. **RMB-click** on the **Translate Y** and choose **Motion FX > Limit**. Set **Minimum** to **0** and **Maximum** to **6**. Now the ball moves flat with some bumps instead of going up and down the whole way.



11 There should not be any noise while the ball is bouncing. It is only needed when the ball is rolling. You can keyframe the **Amplitude** to turn the noise on and off.

In the newly created *motionfx* network, select the *noise1* CHOP node. Go to **Frame 37** where the ball stops bouncing and starts rolling. **Alt-click** on **Amplitude** to set a keyframe. Go to **Frame 1** and set **Amplitude** to **0**. **Alt-click** on **Amplitude** again to set a second keyframe. Go to the **Animation Editor** and select the curve. Click on the **Constant** button in the Functions bar. This creates a sharp cut from no amplitude to an amplitude of 1.



12 When you are finished, close the window then **RMB-click** on the **Motion Path** handle and turn off **Persistent** so that it isn't visible when you deselect.

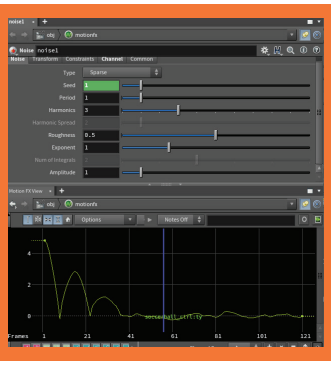
In the toolbar at the left side of the Scene view, click on the **Render Flipbook** button. Leave the default settings and click **Start**. Wait while the sequence is captured and then you will see the flipbook in an **Mplay** window. You can **Play** this back and scrub through to evaluate your motion.

Save your work.

MOTION FX

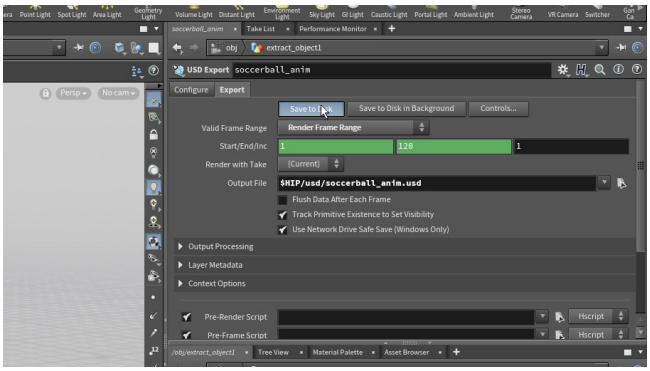
While keyframes and animation curves are stored in the parameters of your nodes, you can also use **channel operators (CHOPs)** for a more procedural node-based approach to working with motion.

Motion FX can be applied to keyframed motion which is extracted and stored in a **Channel CHOP**. You can then apply effects such as **cycle, noise, smooth, limit** or **lag** to the existing motion. On the **Constraints** shelf, you have tools which let you have one parameter either **look at, lag** or **jiggle** behind another.



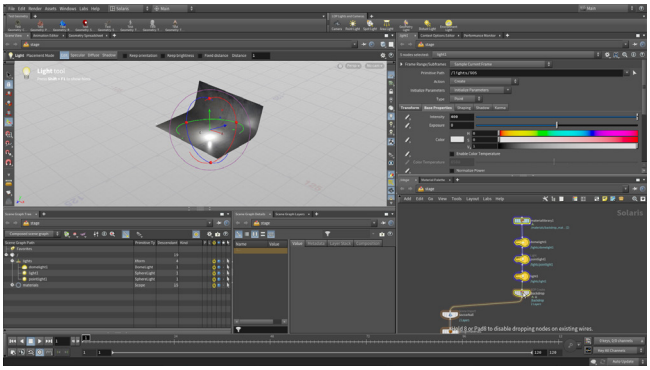
PART NINE
Lights, Camera, Action!

To render out the animated soccer ball, you will need to go back to the Solaris environment and set up a second shot. You will begin by branching off new LOP nodes from backdrop geometry then adjust the lights and cameras to suit the bouncing soccer-ball animation. You will also set up motion blur for the deforming geometry.



01 Go to the Object level. **Double-click** on *soccerball_anim* to dive into it. **Press n** to select all the geometry then go to the **Modify** shelf and select **Extract**. This takes all the motion and bending of the ball and puts it in one network.

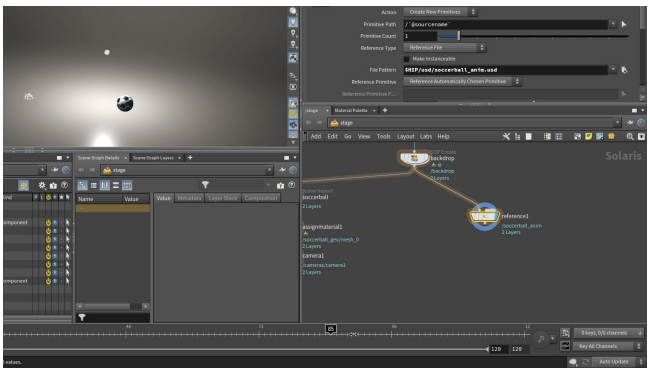
You can see an *objectmerge* node which is extracting the ball into a new object called *extract_object*. **RMB-click** on the output and find **USD Export** then click to place this node. **Rename** this node to *soccerball_anim*. Click on the **Export** tab then set **Valid Frame Range** to **Render Frame Range** and set **Output File** to *$HIP/geo/soccerball_anim.usd*. Click the **Save to Disk** button.



02 Go back to the **Solaris** desktop and point it to */stage*. In the **Network View**, add a **Null** node just before the *karmarendersettings* node and call it *SHOT_01*.

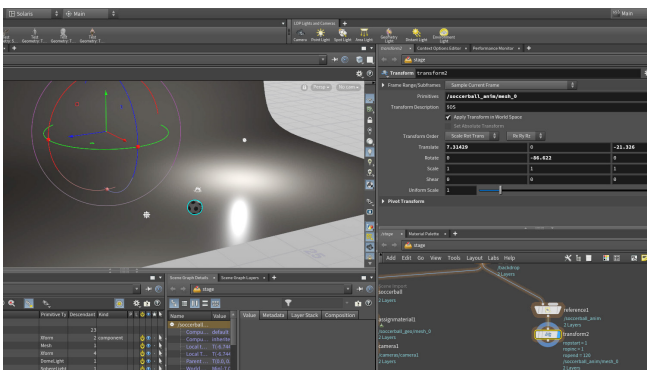
Disconnect the three **light** nodes (not the *lightmixer*) and move them above the *backdrop* node. This will not change how the first shot looks, but will let you share the nodes between the two shots.

Move the *backdrop*, *light* and *materiallibrary* nodes to the right.



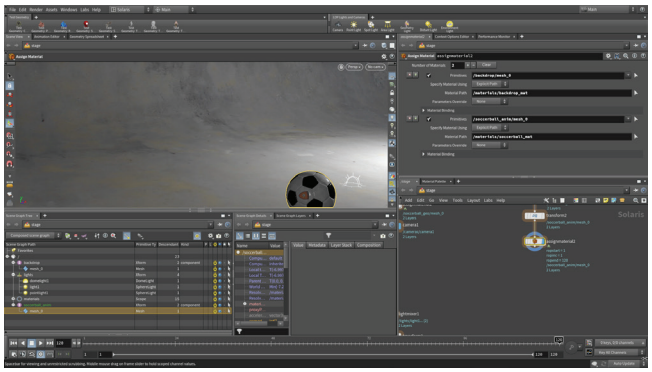
03 Zoom in and add a **Reference** node to the right under the *backdrop* node. Connect the *backdrop* node to this new node and set its **Display Flag**. Next to **File Pattern**, click on the **File Chooser** and find the *soccerball_anim.usd* file. Rename the node to *soccerball_anim*.

Scrub in the timeline, in Houdini GL view, to see the cached animation which is part of the USD file.



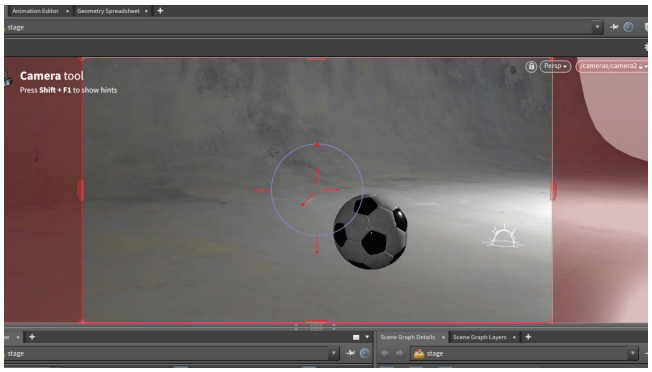
04 Use the **Select** tool and click on the new animated soccerball. In the Scene View, press **tab > Transform** to add a transform node to the graph.

In the Scene view, use the Transform handle to **Move** the Ball to the back of the backdrop in the middle. **Scrub** the timeline to watch the ball bounce to the right. Leave it at somewhere around frame 80. **Press R** to get the rotate handle. **Rotate** to move the ball so that it is bouncing at an angle from the backdrop. **Scrub** the timeline to see if you like its direction.



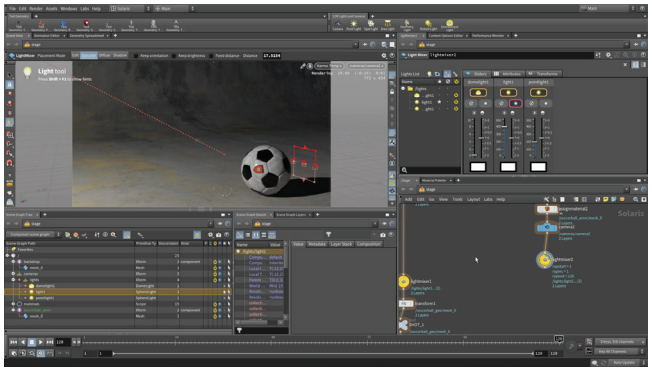
05 In the Network view, **Select** the *assignmaterial* node from the SHOT 1 network then **Alt-drag** to create a copy of this node. Wire the *transform* node into the *assignmaterial* node then set its **Display Flag**. This will assign the material to the *backdrop* but since the soccerball primitive has changed it needs to be reassigned.

In the field next to **Primitives** for the *soccerball_mat* , change the primitive name to */soccerball_anim* to reassign the material to the new geometry.



06 Tumble around until you see the ball animated towards the camera from the top left to the bottom right. In the **LOP Lights and Cameras** shelf, **Alt-click** on the Camera tool to place a camera from the angle you are currently looking.

Press the **Lock Camera/Light to View** button so that view changes can be used to reposition the camera. Now **Tumble**, **Pan** and **Dolly** in the viewport to tweak the camera to get the framing that you want for the shot. Scrub the timeline to make sure the camera works for the whole sequence.



07 Add a **Light Mixer** node after the *camera*. On the *lightmixer* node you will need to move over the lights. This will let you use the same light handles you learned about earlier to make lighting decisions for this shot and use the **Karma** display in the viewport to verify your setup. You can also use the *lightmixer* node to play with **intensity** and **exposure** for this shot.

These edits are being held in the *lightmixer* node and changes are not being made to the original lights. The *lightmixer* lets you tweak existing lights when working in a multi-shot setup.

08 In the Network view, **Alt-drag** the *SHOT_01* and *karmarendersettings and usdrender_rop* nodes. Wire the *lightmixer* node into this chain. Select the new *karmarendersettings* node and make sure that **Camera** is set to */camera2*. Set **Valid Frame Range** to **Render Frame Range** and set the **Output Picture** to *$HIP/render/anim/soccerball_anim_$F2.exr*. The *$F2* adds frame numbers to the renderings with a padding of two and the */anim/* creates a directory to hold these frames.

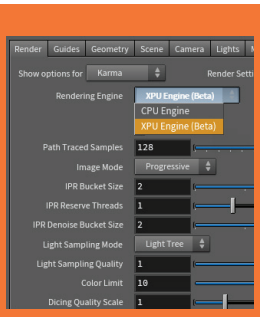
On the *usdrender_rop* node, click **Render to Disk**. When you finish, choose **Render > Mplay > Load Disk Files** and open up the rendered images to review the final sequence. **Save** your work.

KARMA RENDERER

Karma is a physically-based **HYDRA** renderer built to work with **USD** files in the Solaris/LOPS context. This allows it to be used in the **Viewport** for interactive updates or to be rendered to disk using a **Karma** node.

Note: Houdini 19 includes a preview for an upcoming **Karma XPU** render engine. This hybrid GPU/CPU renderer is **Alpha** with many features still under development and is for **testing purposes only**. You can choose **XPU** in the Scene view's **Display Options** or in the **Karma** node.

Viewport Menu

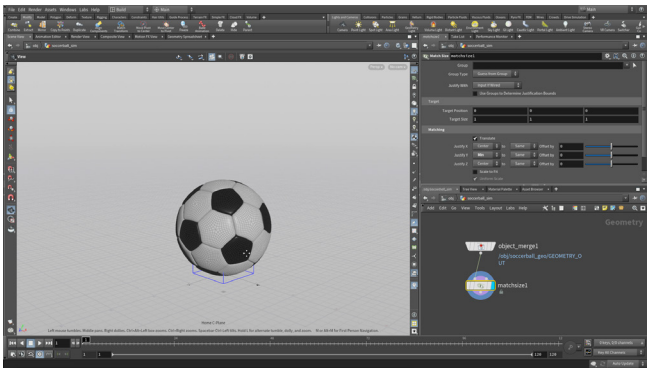


Display Options

PART TEN

Set up a Rigid Body Simulation

While traditional animation is great for animating a single soccer ball, dynamics would be a better option if you want to animate a bunch of soccer balls. Dynamics requires a simulation so that the solver can go frame by frame determining how each of the participating objects interact with each other. You will use packed geometry to get an efficient result for this simulation.

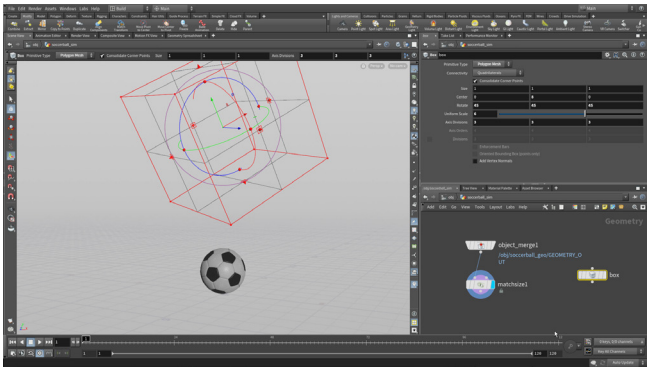


01

Change back to the **Build** desktop and navigate to the object level. Hide all of the animation rig nodes and the *extract_object* node by turning off their display flags. Turn on the *soccerball_geo* display.

Select the *soccerball_geo* node then from the **Modify** shelf click on the **Extract** tool. This creates a new object with the soccerball object merged. Jump up one level and rename *extract_object* to *soccerball_sim*. Hide the *soccerball_geo* object.

Dive back in to the *soccerball_sim* object to work with the geometry. Add a **Match Size** node to center the ball around the origin.



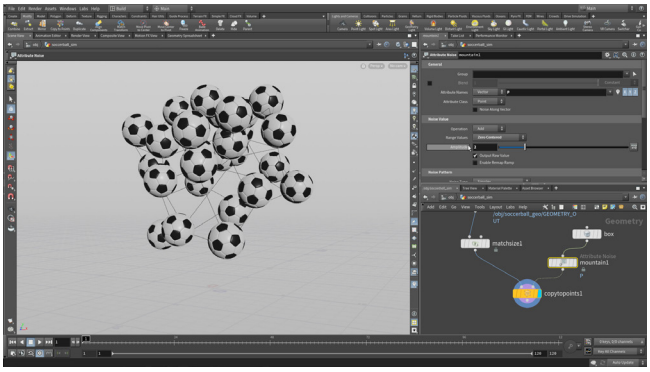
02

In the Network view, press **tab > Box** then place it to the right of the *matchsize* node.

Set the following on the *box* node:

- **Center** to **0, 8, 0**
- **Rotate** to **45, 45, 45**
- **Primitive Type** to **Polygon Mesh**
- **Uniform Scale** to **6**
- **Axis Divisions** to **3, 3, 3**

This puts it in the right position for the simulation.

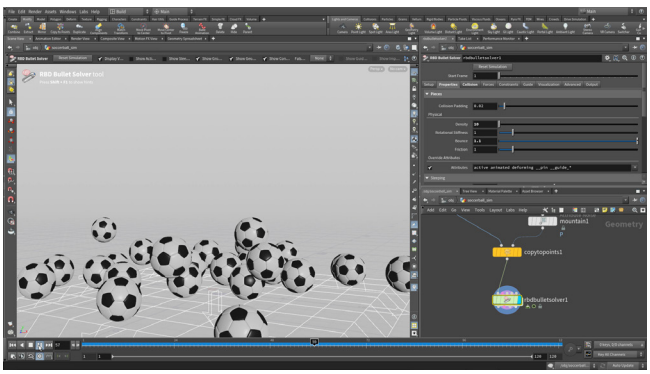


03

In the Network view, add a **Copy to Points** node just below the other nodes. Wire the *matchsize* node into the first input and the *box* node into the second.

Turn **ON** the **Pack and Instance** option. This will create a faster simulation because the geometry is being instanced to the points of the cube. Set the *copytopoints* node's **Display Flag**.

In the Network view, press **tab > Mountain** and place the node between the *box* and the *copytopoint* nodes. Turn **Off** the **Noise Along Vector** option then set **Amplitude** to **2** and **Range Values** to **Zero Centered**. This will jiggle the points on the box.



04

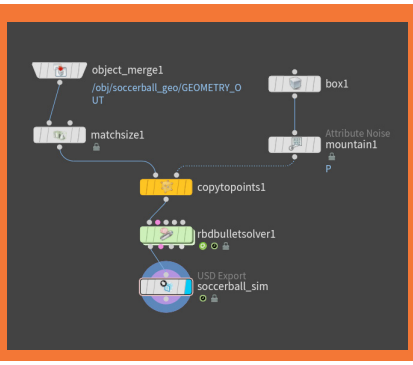
Make sure you are on **Frame 1**. Add a **RBD Bullet Solver** node after the *copytopoints* node. Click on the **Collision tab**, scroll down to **Ground Collision**, and set **Ground Type** to **Ground Plane** to **Ground Plane**. Press **Play** to test out the simulation. The sim is cached which lets you scrub in the timeline to review the results.

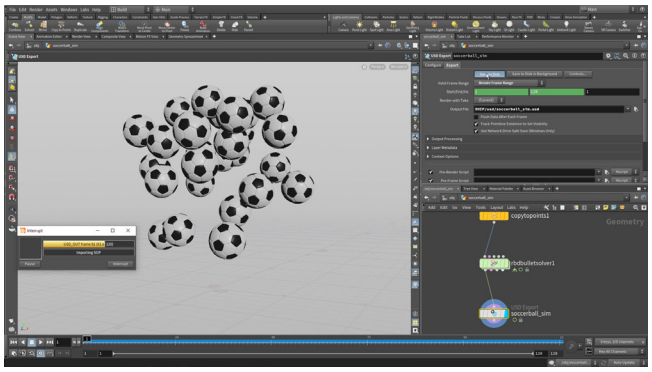
Under the **Collisions** tab, set **Bounce** to **0.8.** Under the **Properties** tab, set **Density** to **10**, **Bounce** to **1.1**. At the top of the Parameter pane for this node, click the **Reset Simulation** button and then press **Play** to resim. Scrub to evaluate.

DOPS hidden inside SOPS

In Houdini, simulations are processed using the **Dynamic Operators** or **DOPs**. With the **RBD Bullet Solver** node in the **Geometry/SOP** context, you are working with a node that has a Dynamics network buried inside it.

This makes it easy to set up at the geometry level with all the DOP nodes are wired up and ready to go but hidden from view. For simpler setups, working at the geometry level will give you a proper simulation. If you need more control over the different solvers then you would need to work directly in DOPs.

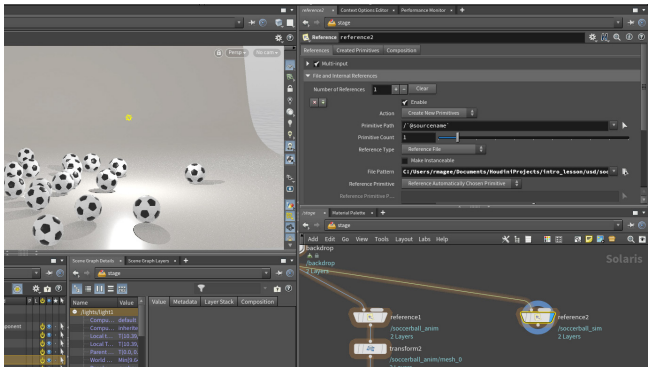




05 At the end of the chain, add a **USD Export** node, set its **Display flag** and **rename** it to *soccerball_sim*.

Set **Valid Frame Range** to **Render Frame Range** and set the **Output File** to *$HIP/geo/soccerball_sim.usd*.

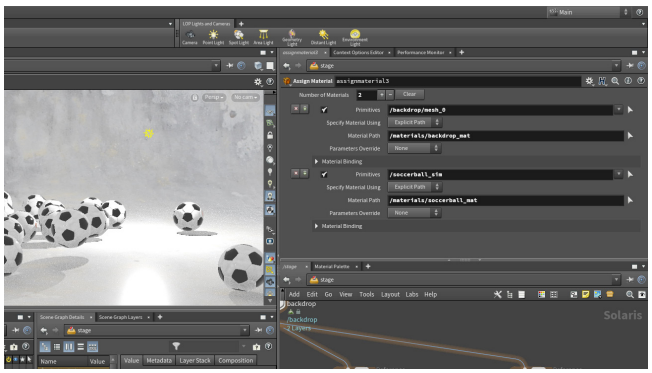
Click on the **Save to Disk** button and this will save the USD file into your geo directory. You will reference this cached asset into the Solaris setup as a third shot.



06 Change your **Desktop** back to **Solaris** and set the path to */stage*. Make sure you are choose **Houdini GL** from the *persp* menu.

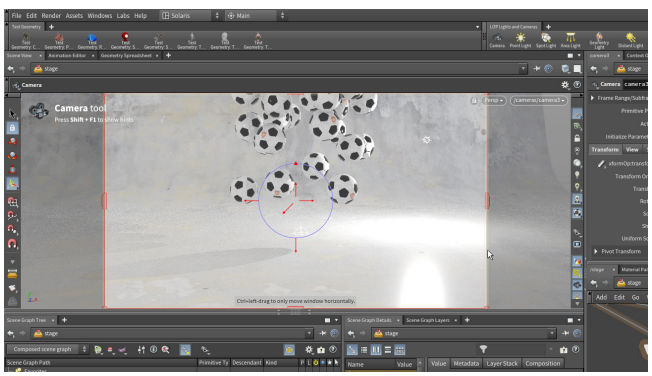
Alt-drag the soccerball_anim **Reference** node and set its **Display Flag**. Set **File Pattern** to *$HIP/geo/soccerball_sim.usd*.

Rename this node to *soccerball_sim*.



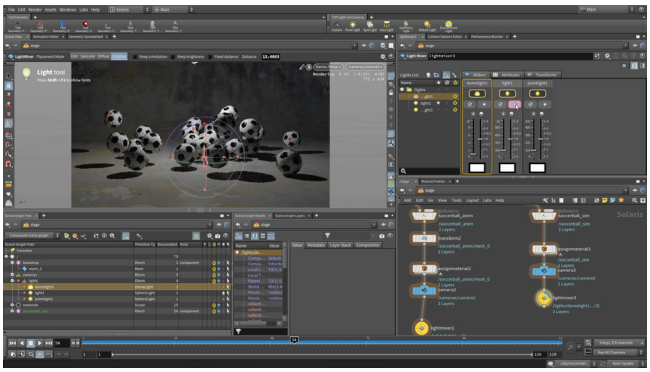
07 In the Network view, **Select** the *assignmaterial* node from the SHOT 2 network then **Alt-drag** to create a copy of this node. Wire the *soccerball_anim* node into it the *assignmaterial* node then set its **Display Flag**. This will assign the material to the *backdrop* but since the soccerball primitive has changed it needs to be reassigned.

In the field next to **Primitives** for the *soccerball_mat* , change the primitive name to */soccerball_sim* to reassign the material to the new geometry.



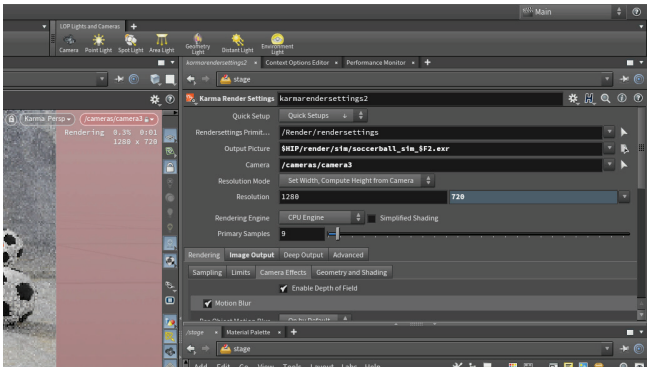
08 Tumble around until you see the balls animated towards the camera. In the **LOP Lights and Cameras** shelf, **Alt-click** on the Camera tool to place a camera from the angle you are currently looking.

Press the **Lock Camera/Light to View** button so that view changes can be used to reposition the camera. Now **Tumble**, **Pan** and **Dolly** in the viewport to tweak the camera to get the framing that you want for the shot. Scrub the timeline to make sure the camera works for the whole sequence.

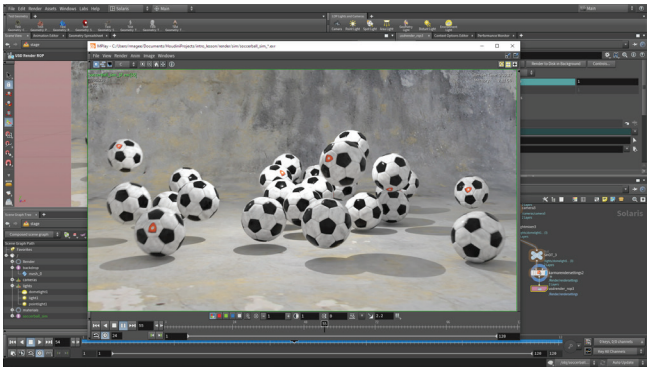


09 Add a **Light Mixer** node after the *camera*. On the *lightmixer* node you will need to move over the lights. This will let you use the same light handles you learned about earlier to make lighting decisions for this shot and use the **Karma** display in the viewport to verify your setup. You can also use the *lightmixer* node to play with **intensity** and **exposure** for this shot.

These edits are being held in the *lightmixer* node and changes are not being made to the original lights. The *lightmixer* lets you tweak existing lights when working in a multi-shot setup.



10 In the Network view, **Alt-drag** the *SHOT_02* and *karmarendersettings and usdrender_rop* nodes. Wire the new *lightmixer* node into this chain. Select the new *karmarendersettings* node and make sure that **Camera** is set to */camera3*.



11 On the *usdrender_rop* node, set **Valid Frame Range** to **Render Frame Range** and set the **Output Picture** to *$HIP/render/sim/soccerball_sim_$F2.exr*. Click **Render to Disk**.

When you finish, choose **Render > Mplay > Load Disk Files** and open up the rendered images to review the final sequence.

CONCLUSION

You have now built a scene from scratch, touching on many different aspects of Houdini. You have modeled, set up textures, animated, rendered and simulated. Along the way you have learned about the different Houdini contexts and how to navigate back and forth between them.

While this lesson doesn't result in blockbuster VFX, it introduces you to fundamental skills which you will carry with you as you dive deeper into Houdini and begin exploring its comprehensive toolset.

There is a wealth of learning material available on the SideFX website to help you take your next steps.

Best of luck on your journey!

HOUDINI FUNDAMENTALS

NODES, NETWORKS & DIGITAL ASSETS

A great way to understand Houdini's node-based workflow is to explore it in the context of a project. It is important to start learning how to think and work procedurally. In this lesson, you will learn how to create your own custom **brickify** tool using procedural nodes and networks to define its function and interface.

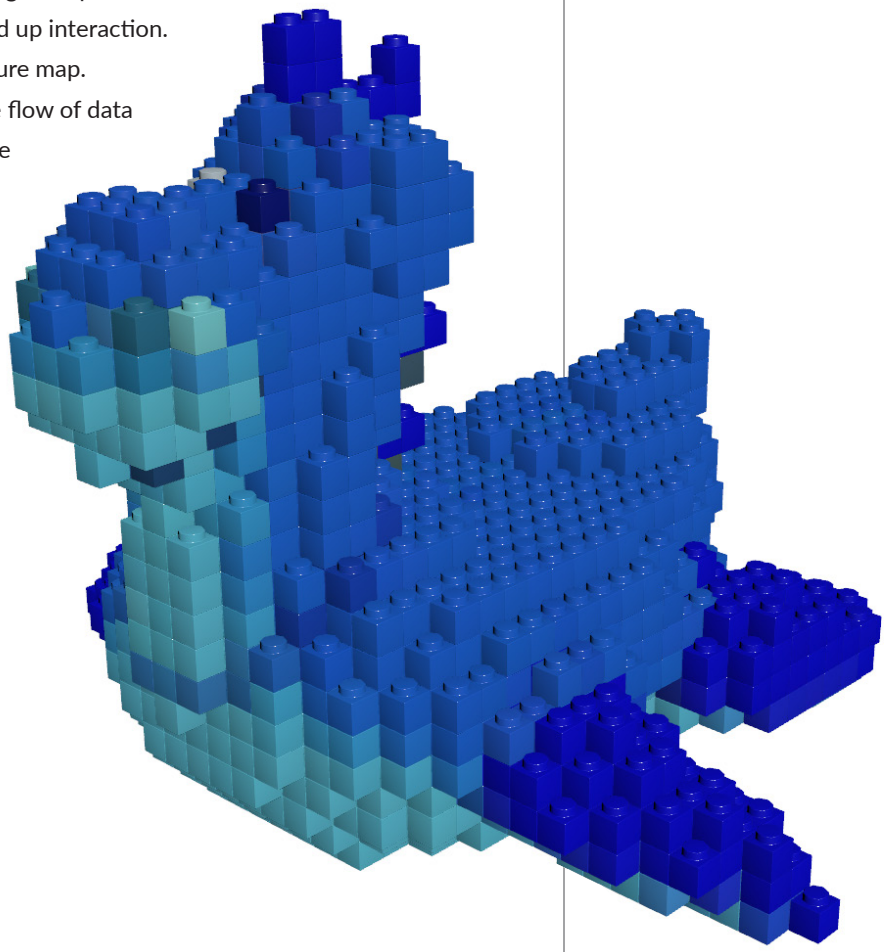
Along the way you will get to use different aspects of Houdini's workspace. Be sure to refer to the overviews in the introduction to remind yourself of how these UI elements work together. The lessons will then give you a chance to put your ideas into practice, which is one of the best ways to learn.

LESSON GOAL

- *To create a custom tool that turns any given 3D shape into toy bricks.*

WHAT YOU WILL LEARN

- How to model a plastic interlocking brick
- How to break down a default rubber toy shape into a grid of points.
- How to use packed primitives and instancing to speed up interaction.
- How to use attributes to color the bricks using a texture map.
- How to work with nodes and networks to control the flow of data
- How to create a Digital Asset to package up and share your solution with others.
- How to animate the bricks appearing over time.



LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

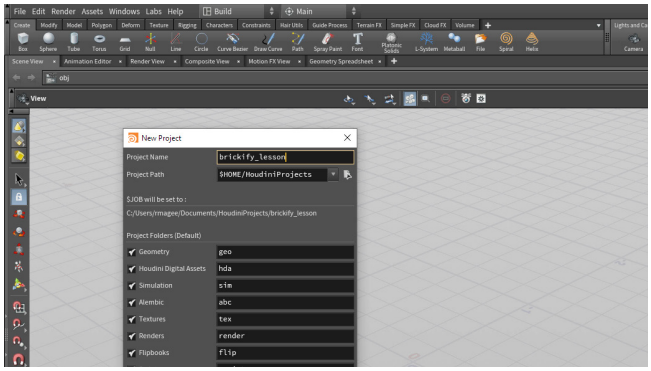
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 3.0 | July 2022

PART ONE

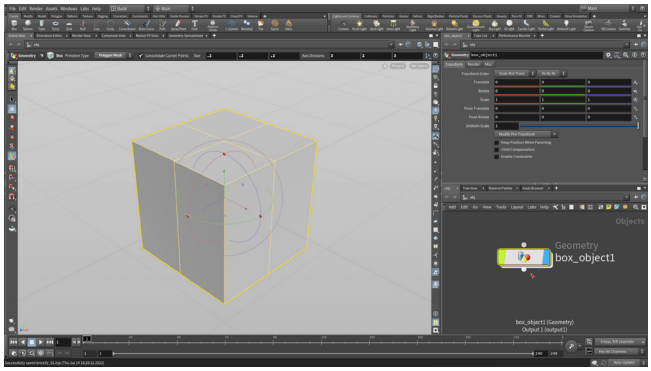
Create a Single Brick

To get started, you will build a single brick model that you will later copy onto points to create the brickified shape. You will create this shape using a combination of polygon modelling tools. Along the way you will see how each action you take creates a node in Houdini that creates a recipe of the steps taken to create the geometry.



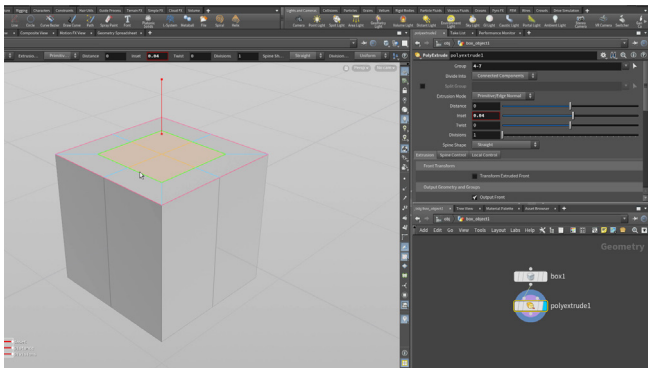
01 Select **File > New Project.** Change the **Project Name** to *brickify_lesson* and press **Accept**. This creates a project directory with subfolders for all the files associated with this shot.

Select **File > Save As...** You should be looking into the new *brickify_lesson* directory. Set the file name to *bricks_01.hip* and click **Accept** to save.



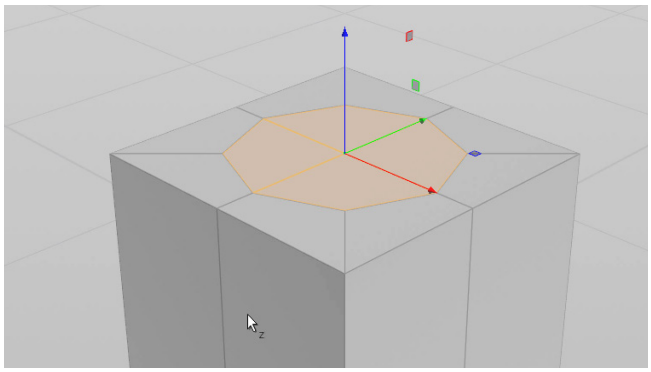
02 In the viewport, **press c** to bring up a radial menu. From this menu, choose **Create > Geometry > Box**. Your cursor now shows the outline of a box waiting to be placed in the scene. **Press Enter** to place it at the origin. In the **Operation Controls** bar, set **Size** to **0.2, 0.2, 0.2** and **Axis Divisions** to **3, 2, 3**.

You can see that there is a *box_object* in the Network view. This object level node contains the transformation information for this shape. The **Operation Controls** show you parameters from another *box* node one level down.



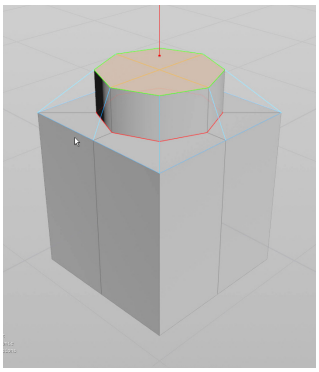
03 Click on the **Select** tool then **Press 4** to go to primitive selection mode. Select the top four faces on the box. **Press c** to bring up a radial menu and choose **Model > Polygons > PolyExtrude**. In the Network view you can see the *box* node feeding into a *polyextrude* node.

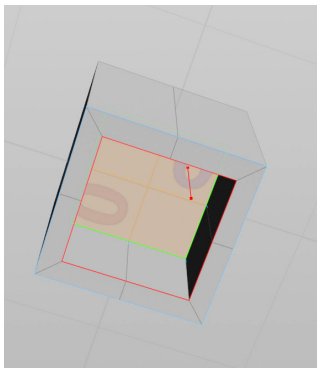
In the Parameter pane, use the slider to set **Inset** to **0.04** which creates new polygons on the top surface of the box. Each node contains parameters relevant to the purpose of that node. These are geometry nodes that are otherwise known as surface operators or SOPs.



04 Next, press the T key to call up the Move tool. This adds an *edit* SOP node into the network. In the viewport **RMB-click** in empty space to bring up a menu and select **Make Circle** to round out the selected polygons.

This menu is associated with the *edit* node. Every node has its own interface that you can access as long as a relevant tool is active. In this case, the Move tool gives you access to the handle. In other cases the **Handle** tool would be used to access the interactive handles for a node.



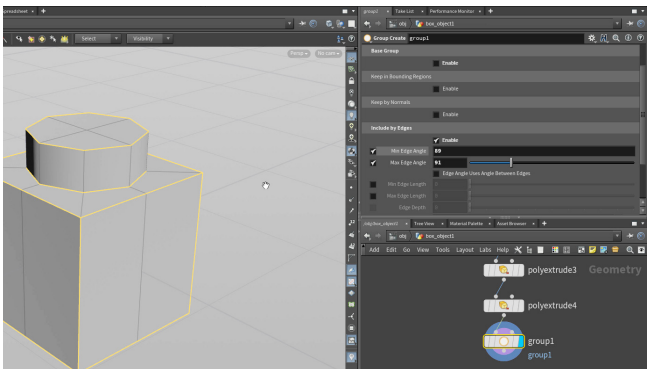


05

Press c to bring up a radial menu and choose **Model > Polygons > PolyExtrude**. In the Scene View pane, use the handle to drag up the polygons and set **Distance** to **0.05.**

Tumble around and **press s** to go into select mode then select the bottom four polygons of the box. **Press q** to repeat the **PolyExtrude** tool. In the Parameter pane, use the slider to set **Inset** to **0.025.**

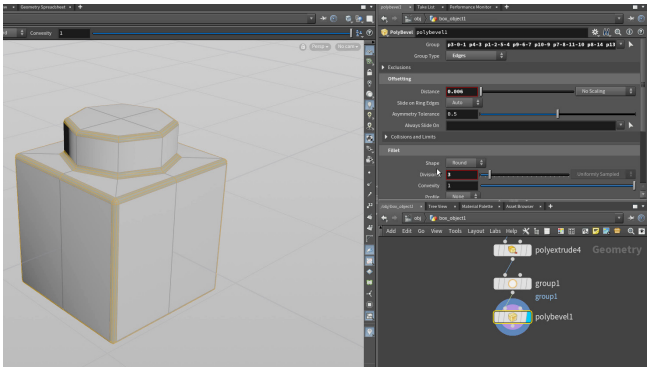
Press q to repeat that tool and set a **Distance** of **-0.175.** When you finish, tumble back to see the top of the brick.



06

Press 3 to go to edge selection and press n to select all the Edges. In the Scene view press tab and start typing **Group...** Select **Group** and in the Parameter pane, set the **Group Name** to *bevel_edges*.

Next, set **Enable** to **OFF** under **Base Group** and then set Enable to ON in the **Include by Edges** section. Turn on **Min Edge Angle** and set it to **89** and then turn on **Max Edge Angle** and set it to **91**.

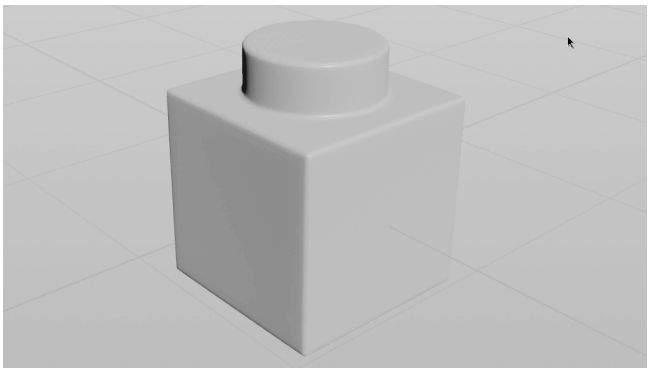


07

Press s to go to the **Select** tool. **Press 9** to turn on the "**Select Groups**" option. In the popup window, click on the *bevel_edges* group.

In the viewport, **press c** to bring up a radial menu. From this menu, choose **Model > Polygons > PolyBevel**. This adds a polybevel node and automatically fills in the **Group** field with *bevel_edges*.

Now set the **Bevel Offset** to **0.006**. Under **Fillet**, set **Shape** to **Round** and **Divisions** to **3**.



08

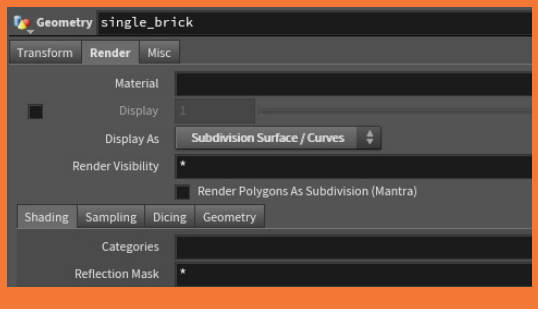
Go to the object level and in the Network view, rename the object to *single_brick*. With the brick selected, press **Shift +** to turn on subdivision surface display for this shape. Deselect the brick to see the model subdivided. If you see wire lines on your object, **press v** and from the radial menu, choose **Shading > Smooth Shading** to hide them.

Save your work.

SUBDIVISION DISPLAY

You can use **Shift +** and **Shift -** to turn subdivision display on and off on selected polygonal objects. This creates a viewport subdivide that lets you see what the shape would look like subdivided. These hotkeys set the **Display As** parameter which can be found at the object level on the object's **Render** tab.

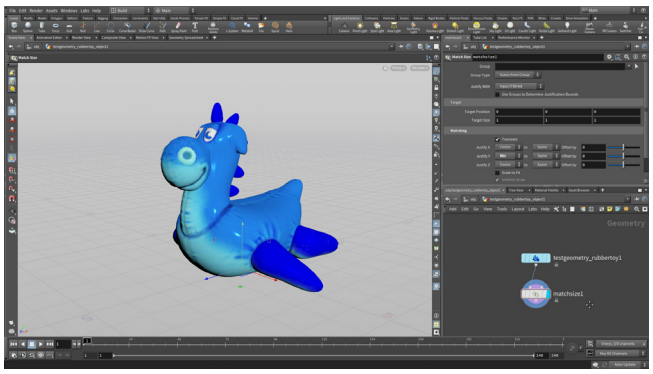
Your object will not render with subdivisions unless you turn on **Render Polygons As Subdivision Surfaces** on the same tab.



PART TWO

Copy Bricks to a Point Cloud

You are now going to create a cloud of points that match the shape of a particular piece of geometry. You are then going to instance the bricks to the 3D grid to create a brickified version. The instancing is generated by packing the brick geometry then instancing them to the points.

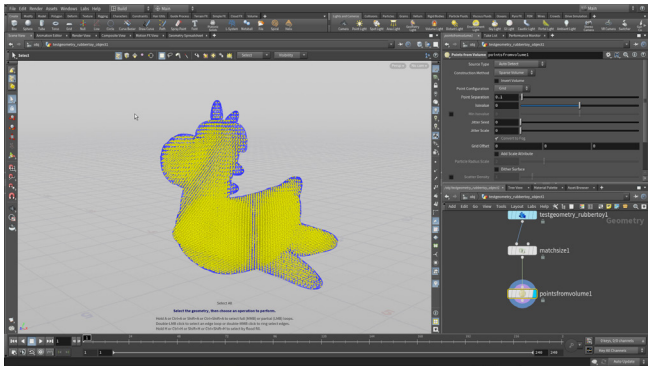


01 In the Scene View, press tab and start typing **Test...** then choose **Test Geometry: Rubber Toy.** Press **Enter** to place it at the origin.

Press i to dive into the *test geometry* object. Set the following:

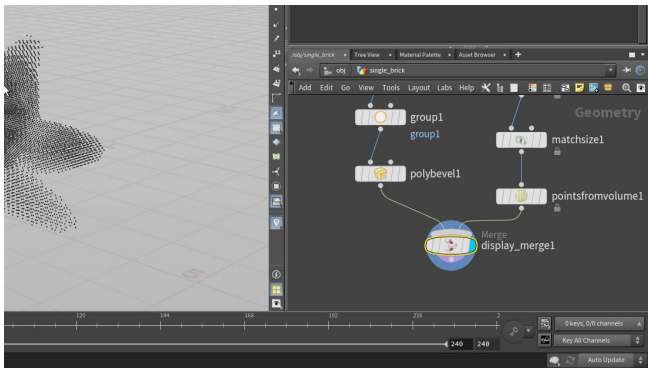
- **Uniform Scale** to **3**

Now add a **Match Size** node and set **Justify Y** to **Min**. This raises the toy so that it sits on the ground.



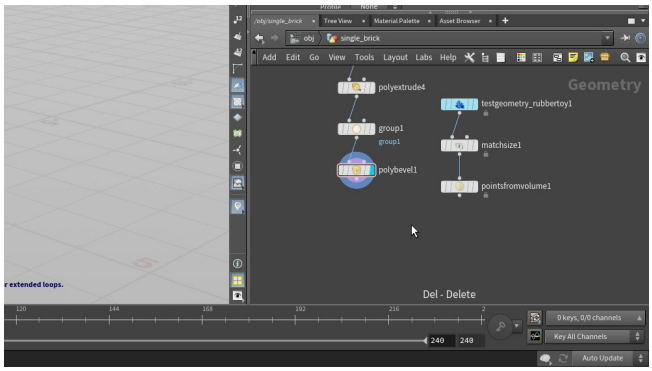
02 In the Network editor, **RMB-click** on the output of the *matchsize* node and type **Points...** then select **Points from Volumes** and place it's node in the network. Now set its **Display flag** to focus on the output of this new node.

Press s to go to the select tool and **press 2** to get point selection and then **press n** to select all the points which will highlight in yellow. You will copy the bricks to these points.

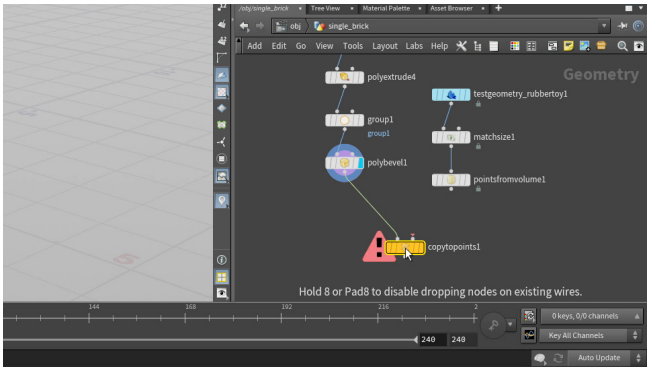


03 Press **u** to go back to the Object level and shorten the name of the rubber toy to *rubbertoy*. In the Network view, press the **shift** key and click on the *rubbertoy* and then the *single_brick*.

From the **Modify** shelf tab, select **Combine** to bring these objects together. You are taken to the geometry level and the nodes are feeding into a merge node.



04 In the Network view, select the *display_merge* node and press the **delete** key. Now the *polybevel* node chain is displayed and the other chain is not. Houdini lets you choose which node you want to display which would be the node that will be visible when you go back to the object level.

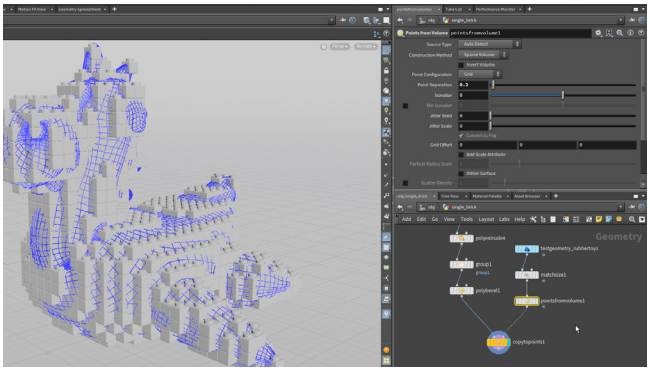


05

RMB-click on the output of the *polybevel* nod, start typing **copy...** and select **Copy to Points** node. Click to place the node below the two chains and set its **Display Flag**.

Turn on the **Pack and Instance** option. This is important because it will display the copied bricks much faster than if this option is off.

At this point there is an error on the node because we haven't connected the second input.

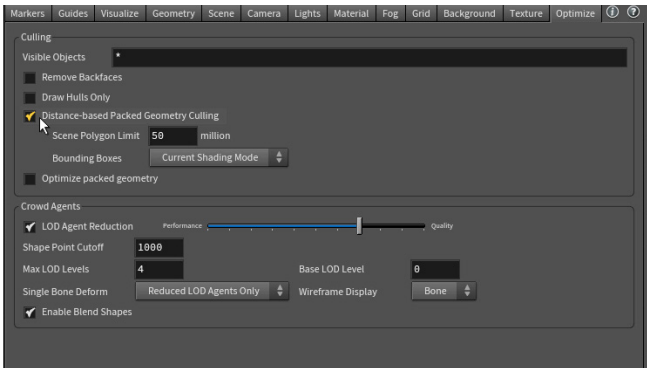


06

Click on the dot under the *pointsfromvolume* node then connect it to the second input on the *copytopoints* node.

The bricks appear to be overlapping. Go back to the *pointsfromvolume* node and set **Point Separation** to **0.2.**

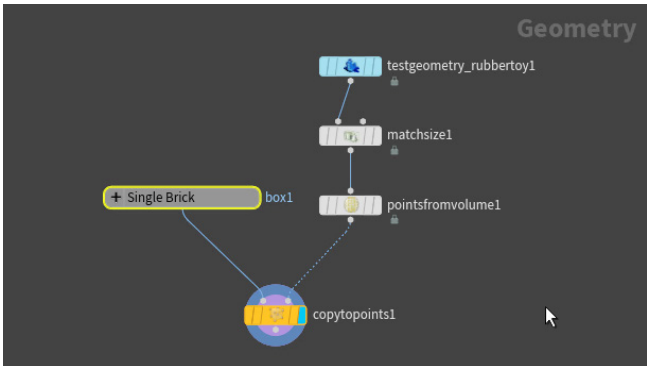
Now you are copying bricks to the points of the grid.



07

Click on the *copytopoints* node. Some of the bricks may appear dark grey and don't display the proper brick. This is because of geometry culling in the viewport. You can fix this with a display setting.

In the Scene View, press **spacebar-d** to bring up the **Display Options.** Click on the **Optimize** tab and either set **Scene Polygon Limit** to a number higher than **50 million** or set **Distance-based Packed Geometry Culling** to **OFF**. Close the floating panel. Now you can see all of the bricks being copied to the points as Packed Instances.



08

Before saving your work, let's Organize the network. Select the nodes that make up the single brick and press **Shift-O** to create a network box around them. Click on the title bar of the box and enter *single brick*. You can then collapse the box and move it down to de clutter the network a bit.

Save your work so far.

PACKED INSTANCES

If you leave **Pack and Instance** setting **OFF** on the copy to points node then you will end up with a large model with over a million points and primitives. This would make it very slow to manipulate in the Viewport because instancing is not being used.

If you turn it **ON** then the **338 points** on the brick model are packed up and instanced which leaves a much more efficient point count for the copy to points node.

Points	1,024,816	Center	0, 2.025, 0
Primitives	1,018,752	Min	-3.1, -0.5, -2.7
Vertices	4,075,008	Max	3.1, 4.55, 2.7
Polygons	1,018,752	Size	6.2, 5.05, 5.4

PACK AND INSTANCE | OFF

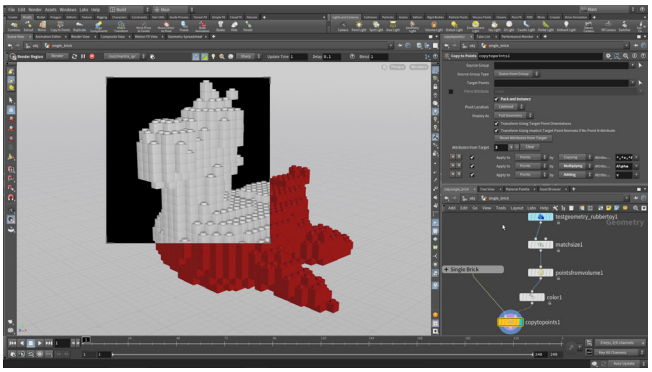
Points	3,032	Center	0, 2.025, 0
Primitives	3,032	Min	-3.1, -0.5, -2.7
Vertices	3,032	Max	3.1, 4.55, 2.7
Packed Geos	3,032	Size	6.2, 5.05, 5.4

PACK AND INSTANCE | ON

PART THREE

Add Color and Switch to a Teapot

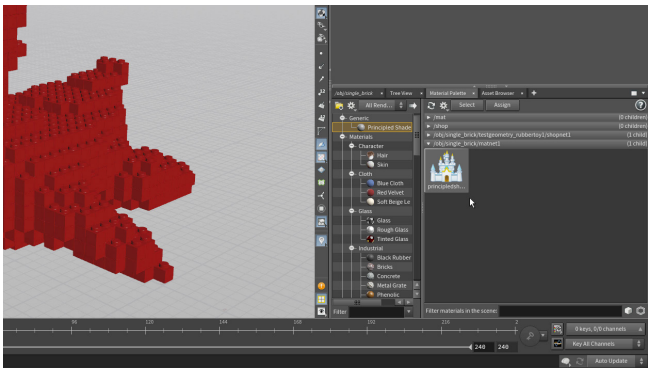
You are now going to add color to the points which will then be picked up by the instanced bricks. At first this transfer of color only applies in the viewport but setting up a proper material lets you set up the point colors for rendering the bricks. You will then set up a switch between the rubber toy and a teapot to make sure that your network works with different shapes.



01 Add a *color* node between the *pointsfromvolume* node and the *copytopoints* node. This will add color to the points which will get copied to the bricks. Change the **color** to **red** to help the bricks stand out against the background.

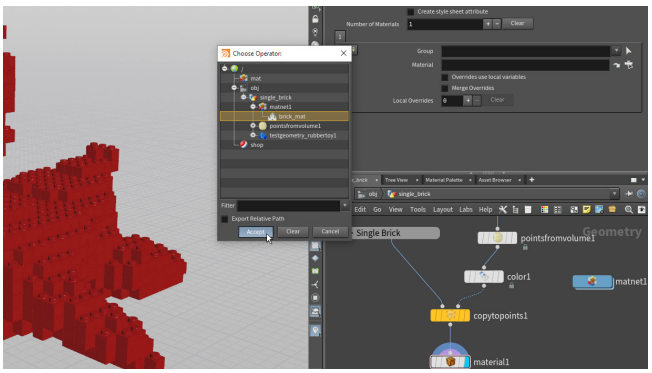
Click on the **Render Region** tool and drag a bounding box over the rubber toy. This kicks off a test Render and you can see that the bricks are not rendering as red. You need a material assigned that picks up the color.

Click the **x** button in the top right of the render region to close it.



02 Press **tab** in the Network view and type **Mat...** Choose the **Material Network** tool and click to place the node down in the network.

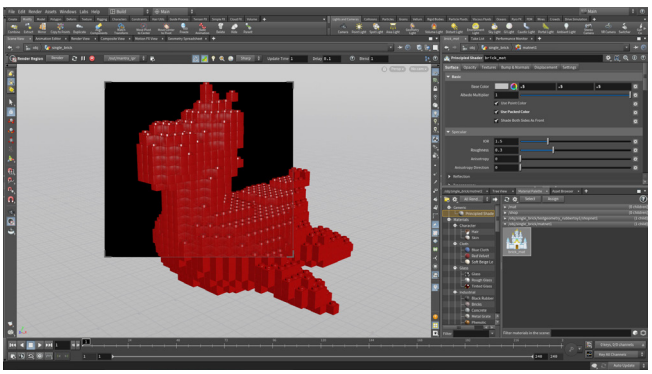
Go to the **Material Palette** and close */mat* and open up */matnet*. Drag a principled shader down into the *matnet*. Name it *brick_mat*.



03 Go back to the geometry network using the back button in the Network view. **RMB-click** on the output of the *copytopoints* node and type **Material...** Select **material** and place it's node in the network and set its **display flag**.

Click on the **Operator Chooser** button on the far right of the **Material** parameter. From the pop-up window, navigate to and highlight *brick-material*. Turn on the **Export Relative Path Option** and click **Accept**.

The material is assigned but the bricks still don't render.

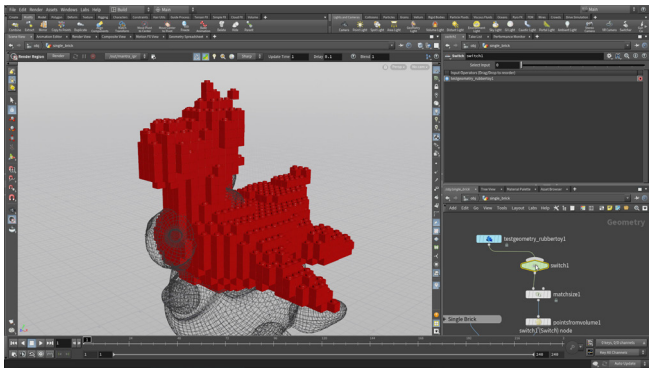


04 Go back to the Material Palette and select the *brick_material*. Click on the **Surface** tab then set the **Base Color** to **0.5, 0.5, 0.5** and turn **ON** the **Use Packed Color** option.

Click on the **Render Region** tool and drag a bounding box over the rubber toy. The rendered bricks should now be red.

If not then click **Render** in the top bar of the Scene view to make sure the changes have been applied.

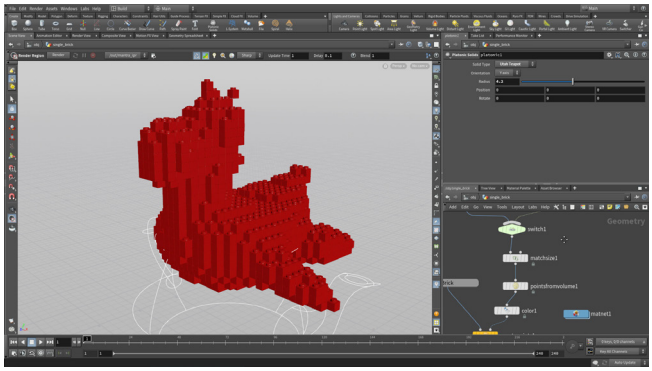
Save your work so far. Click the **x** button in the top right of the render region to close it.



05 Go back to the geometry network using the back button in the Network view. **Press s** to go to the select tool. In the Network view, press **tab** and type the word **Switch** then from the **Sourcing** folder drag it into the Network view.

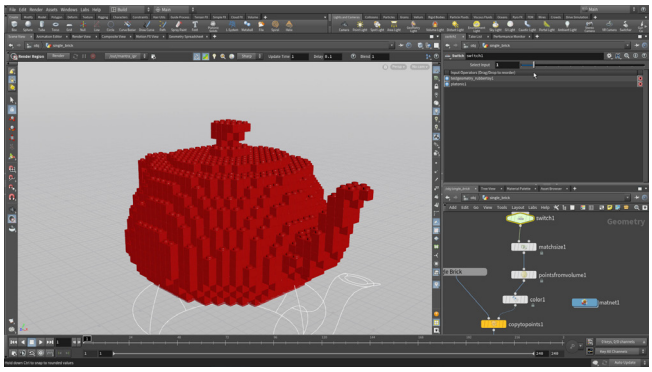
Next, drag it over the line connecting the *rubbertoy* node to the *matchsize* node. This inserts it into the network between the two nodes.

This node will make it easier to switch between different incoming shapes.



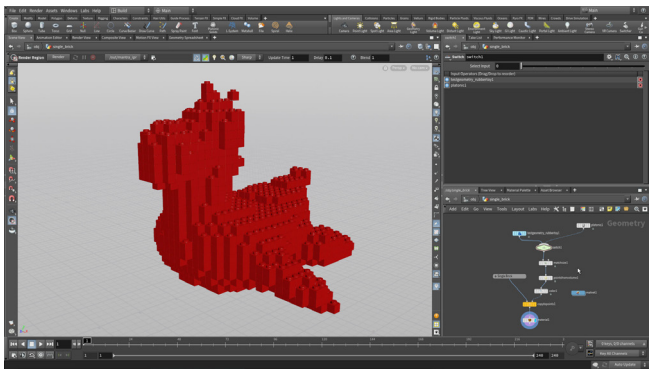
06 In the tool shelf, go to the **Create** menu and drag the **Platonic** tool down to the network view. This places a platonic node at the geometry level. Tools can be dragged into the network view as long as the node type makes sense for your current network level.

Set the platonic node's **Solid Type** to **Utah Teapot**. Set **Radius** to **4.2**. The volume from points node will update the points to match the new volume to generate a different configuration of bricks.



07 Connect the output of the *platonic* node to the input of the *switch* node. Now you can select the *switch* node and change its **Select Input** to **1**. The platonic shape has been cubified using the same setup as the rubber toy.

This is one of the big benefits of a procedural system. Later you will package up this network into a custom tool called a digital asset. As a **digital asset**, it will be easier to share the network with others and to manage the tool as it gets deployed in a studio environment where changes and updates are inevitable as the tool evolves.



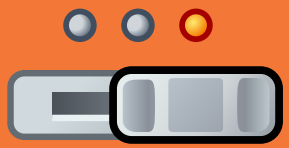
08 Use the *switch* node and set the **Select Input** to **0** to go back to the *rubbertoy*. You can now go back and forth between these two shapes and even add more shapes to see them brickified.

Save your work.

SWITCH NODE

The **Switch** node is a great tool for providing options within a node network. This node lets you quickly explore multiple options without the need to wire and unwire different chains.

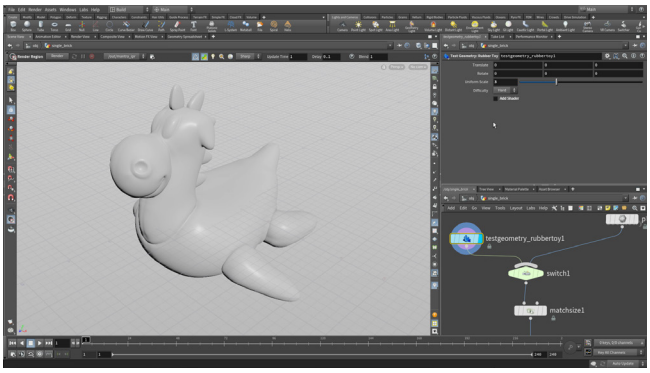
This node is also very useful when you wrap up your network into a **Digital Asset** because you can promote the switch to the asset as either a menu or a slider for quick access to different options.



PART FOUR

Color the Points using a Texture

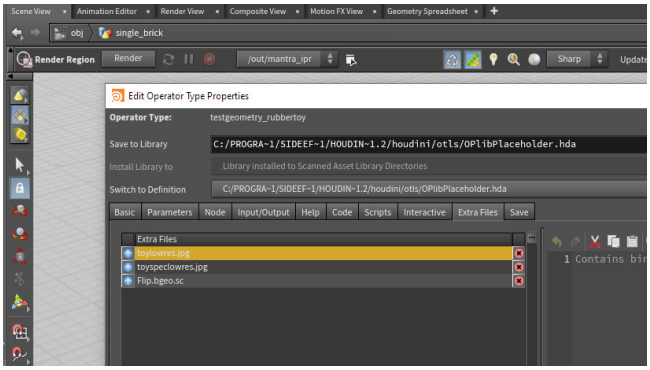
Earlier you added a color attribute to the points which affects the coloring of the bricks instances. Instead of using a single color, you are now going to use a texture map to create a more interesting look for the bricks. This will involve some special nodes to turn the texture into point colors.



01 Turn on the **Display Flag** on the *testgeometry_rubbertoy*. Select the *rubbertoy* node and in the Parameter pane turn **Off** the **Add Shader** parameter.

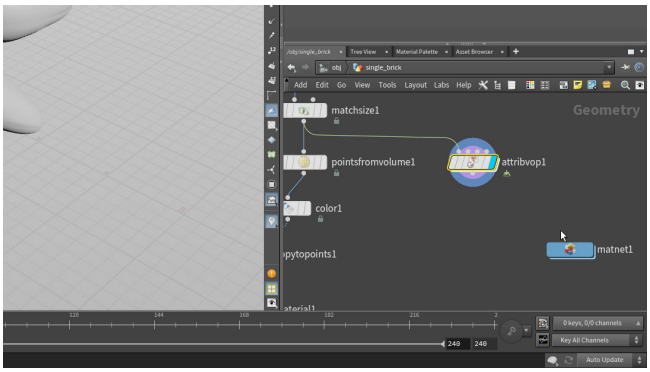
To hide the UV display in the perspective view, go to the **Display Options** bar and turn **Off** the **Show UV Texture when UV's Present** button.

You will bring back the color on the bricks using a different method that involves pulling the color from a Texture Map on disk.



02 From the **Asset** menu, choose **Edit Asset Properties > Rubber Toy**. In the **Properties** window, click on the **Extra Files** tab and select *toylowres.jpg*.

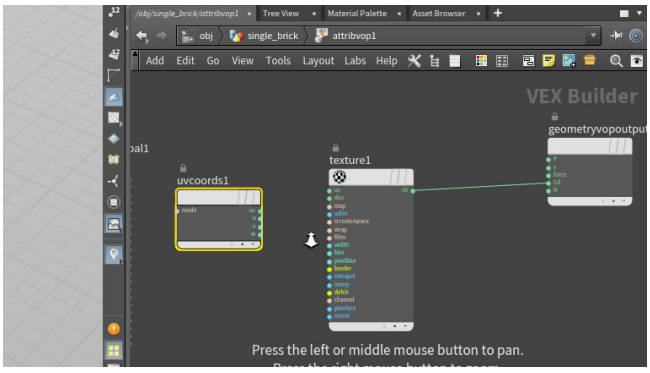
Click the **Save as File** button and save it into the *tex* folder. The texture was stored in the digital asset so that it could be shared along with the asset. You will use the texture on disk to add color to the bricks.



03 In the Network view, press **tab** and start typing **Attribute...** Select the **Attribute VOP** node and place it into the network beside the *pointsfromvolume* node.

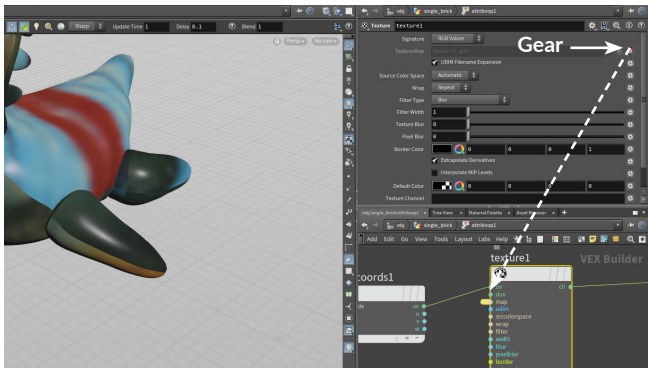
Feed the output of the *matchsize* node into the first input of the *attributevop* node. Set the **Display Flag** on this new node.

In the Parameter pane, set **Run Over** to **vertices**.



04 **Double-click** on the *attributevop* node to dive down and use the **tab** key to add a **Texture VOP**. Feed it into the **Cd** input of the *geometryvopoutput* node.

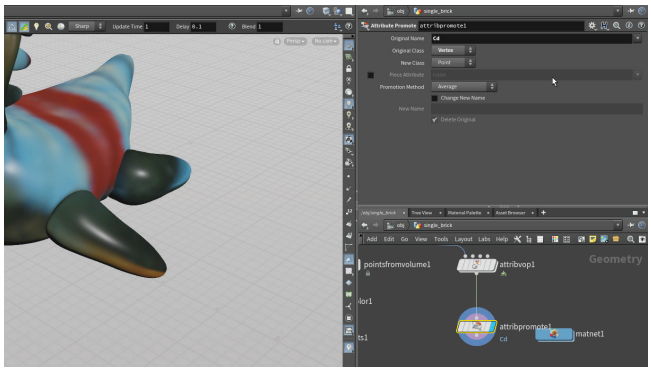
Add a **UV coordinate** node and feed it into the **UV** input on the *texture* node.



05

Select the *texture* node. Click on the **Gear** icon on the far right of the **Texture Map** parameter and from the menu, select **Promote Parameter**. This will add this parameter to the upper level of this node.

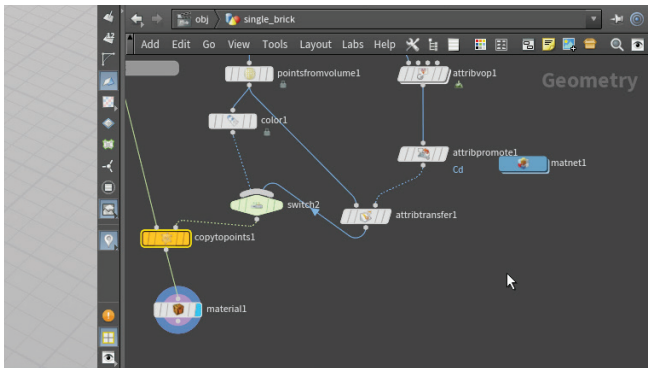
Click on the little knob that appears next to map. In the parameter pane, change the **Label** to **Texture Map.**



06

Press u to go up one level. You can see the **Texture Map** parameter. Leave it set to the default **Mandril.pic** texture for now. You will add the *toylowres* texture map at the end.

Add an **Attribute Promote** to the end of this chain. Set **Original Name** to **Cd** and **Original Class** to **vertex**. Leave **New class** set to **Point**.

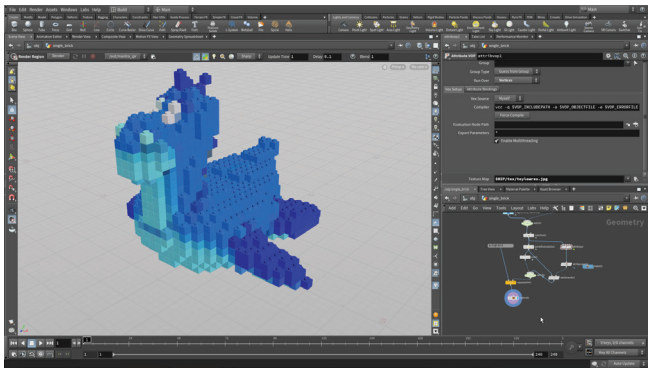


07

Add an **Attribute Transfer** node. Wire the *pointsfromvolume* into the first input and *attribpromote* into the second. In the **Attributes** tab click on arrow on the right side of the **Points** field and choose **Cd**.

Add a switch node after the *color* node and wire the *attribtransfer* node into the switch. Rename this node *texture_switch*.

Set the **Switch** to **1**. Set the **Display Flag** on the *copytopoints* node. Now you can see the colors from the texture map being transferred to the copied points but the bricks are rotated. Turn off **Transform Using Target Point Orientation** to straighten up the bricks.



08

Set the **Display Flag** on the *material* node.

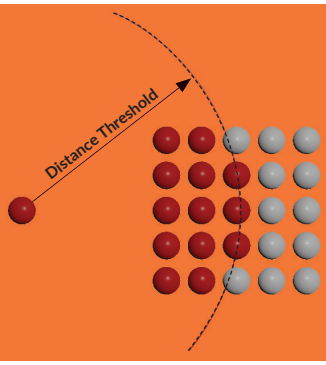
Select the *attribvop* node and click on the **File** button on the far right of the **Texture Map** parameter and navigate to the *toylowres.jpg* texture and click **Accept**. You can see that the texture map colors are now being assigned to the vertices.

Save your work.

ATTRIBUTE TRANSFER

When you want to get attributes from one piece of geometry to another, you can use Attribute Transfer which uses a **Distance Threshold** along with other parameters to get the attributes copied over.

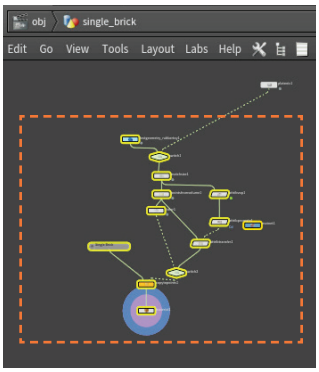
In the case of the rubber toy, you are transferring the **Cd** attribute from the points on the geometry to the point cloud that is used to copy the bricks. You could also copy attributes such as UVs or Capture Weights.

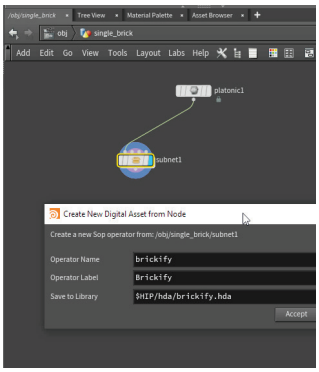


PART FIVE

Create a Brickify Digital Asset

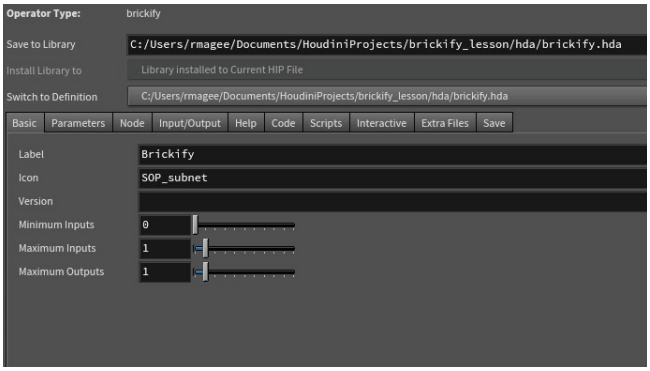
Now that the brickify recipe is working and the nodes are wired together properly, you are going to wrap up some of the nodes to create a single Houdini Digital Asset [HDA] node. Now you can share the network with parameters from inside the asset promoted to the top level to create an interface that can generate unique results each time the asset is used.





01 In the Network view, drag the *platonic* node off to the side. Select all the other nodes in the network and from the **Assets** menu, select **New Digital Asset From Selection....** This will collapse them into a single node.

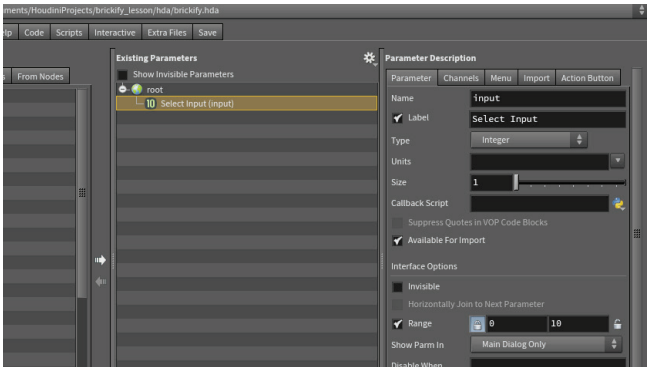
Set the **Operator Name** to *brickify* which in turn changes the **Operator Label** to *Brickify*. Click on the button on the far right of **Save to Library**. Select $HIP in the Location sidebar and then double click on the *HDA* directory. Press **Accept** then click **Accept** again in the **Create New Digital Asset** dialog.



02 This brings up the **Type Properties** window and make sure the **Basic** tab is visible. Set **Minimum Input** to **0** so that you don't "require" an input for the asset to work. The **Maximum Inputs** parameter is set to 1 which defines how many inputs nodes you will allow.

This is the input that is currently connected to the platonic node. When you use this digital asset later, you will use this input to point it to a different shape.

Press **Apply**. **DON'T** press **Accept** because it will close the window.



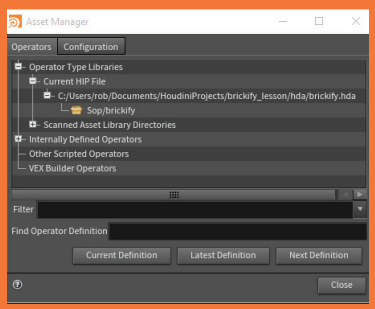
03 In the Network view, **rename** the new asset node *brickify_asset* and **double-click** to dive into it. Click on the *switch* node that is switching between the *testgeometry_rubbertoy* and the *Subnetwork Input* node. The input node is bringing in the platonic teapot shape from the level above.

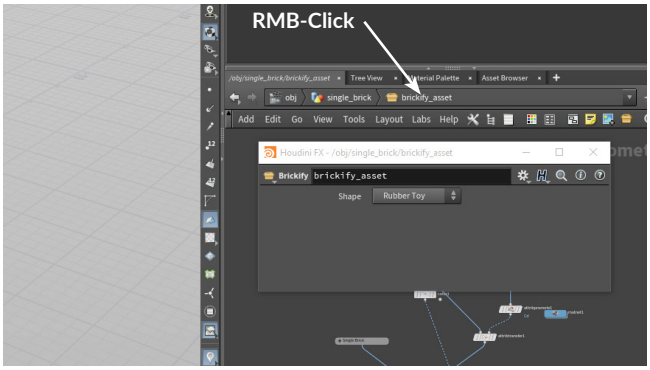
In the **Type Properties** window, click on the **Parameters** tab. Click on the **Select Input** parameter name and **LMB-drag** it to the **Existing Parameters** list in the **Type Properties** window. Drop it on the root to add it to the UI. Click **Apply** which adds the parameter but doesn't close the Type Properties window.

WHAT IS AN .HDA FILE?

When you saved your asset, it creates a **.hda** file on disk. HDA stands for **Houdini Digital Asset** and the asset definition is stored in this file then referenced into your scene. This file contains information about the nodes, the promoted parameters, UI elements and more. This file can be referenced by multiple people into different shots for a shared experience.

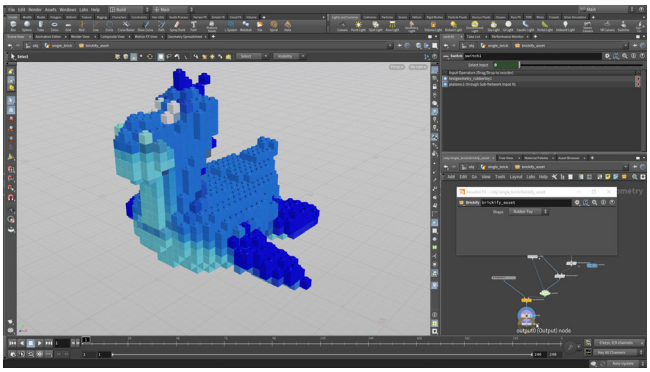
The assets being referenced into your scene can be managed by going to the **Assets** menu, selecting **Asset Manager...** and opening **Current HIP File** .





04 **RMB-click** on *brickify_asset* in the Network View's path bar and choose **Parameter and Channels > Parameters**. Now you have a floating Parameter window with the two *brickify* asset parameters. These are the parameters that will be available to anyone who uses this asset. Let's add some more.

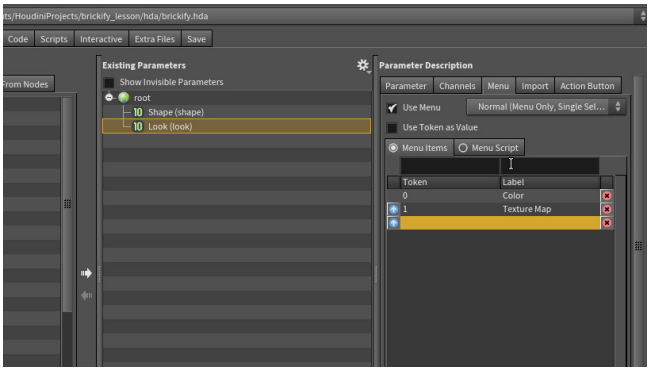
The *brickify* node has a new parameter. Change its value from **0 to 1** and back to see how it affects your scene. The problem is the name isn't very appropriate and a menu would work much better than a slider in this case, so you are going to refine the UI using the Type Properties.



05 In the parameter list, click on the *Select Input* parameter. To the left are options for refining how people see it. Change its **Name** to *shape* and its **Label** to *Shape*.

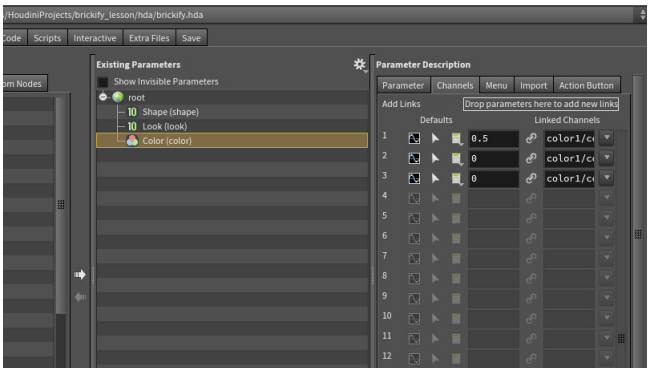
Now click on the **Menu** tab and turn on **Use Menu**. Now under menu items, type **0** under **Token** and *Rubber Toy* under **Label** then press enter. Next type **1** under **Token** and *Custom Shape* under **Label**. Press **Apply**.

Now in the floating parameter pane, you see a **Shape** parameter with a menu. Try it out to see how it works.



06 Select the second *switch* node that sits just under the *color* and *attributetransfer* nodes and promote the **Select Input** parameter to the parameter list. Change its **Name** to *look* and its **Label** to *Look*.

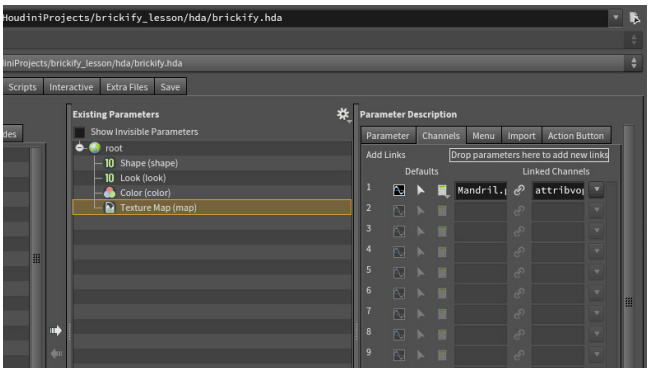
Now click on the **Menu** tab and turn on **Use Menu**. Now under menu items, type **0** under **Token** and *Color* under **Label** then press enter. Next type **1** under **Token** and *Texture Map* under **Label**. Press **Apply**.



07 Now select the *color* node in the network and promote the **Color** parameter to the parameter list. This brings over the color widget and the color wheel as well as the RGB fields.

Press **Apply** in the Type Properties window to see what this parameter looks like in the *brickify_asset* parameter interface.

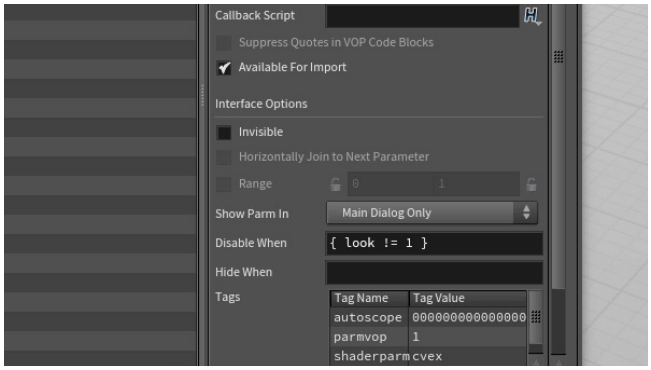
Note: If you press **Accept** by mistake the new parameter is saved to the asset but you will need to use the **Asset** menu and choose Open **Edit Asset Properties... > brickify** to reopen it.



08 Now select the *attributevop* node and promote the **Texture Map** parameter to the parameter list.

In the Parameter description section, click on the **Channels** tab and change the **default** value to **Mandril.pic**. This is a default texture map that doesn't require a path to load and is a more reliable default. Later you will reload the *toylowres.jpg* texture map.

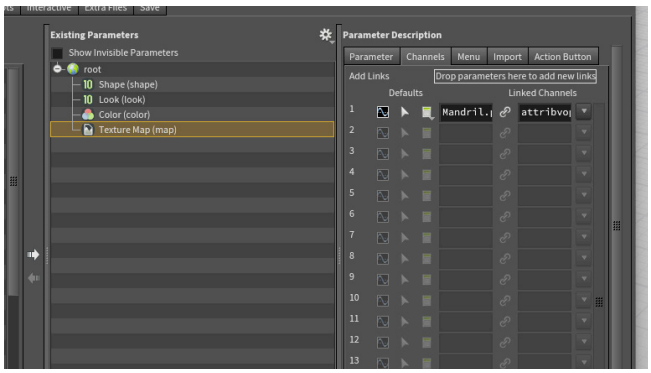
Press **Apply** in the Type Properties window and you will see this parameter in the *brickify_asset* parameter interface and the Mandril texture map is now coloring the bricks.



09 To make it clear which parameters are associated with which shape, you can disable and enable them based on the menu choice. Click on the **Color** parameter and in the **Disable When** field, enter `{ look != 0 }`.

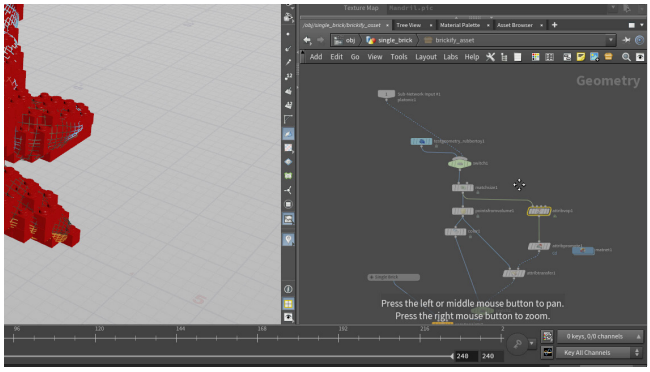
This tells this parameter to disable whenever **Color** has not been chosen in the Look menu. Next, click on **Texture Map** and in the **Disable When** field enter `{ look != 1 }`.

Press **Apply** and test the results using the **Shape** menu. You can see the parameters disabling when they are not needed. You could also hide them with the **Hide When** option but disabling is fine for now.



10 The *brickify_asset* **texture map** parameter now defaults to **Mandril.pic** which is a more versatile default because it is a texture map that will always be available. To go back to the toy map, you would need to click on the file selector next to **Texture Map** and again click on *$HIP* then dive into the *tex* directory and select the *toylowres.jpg* file.

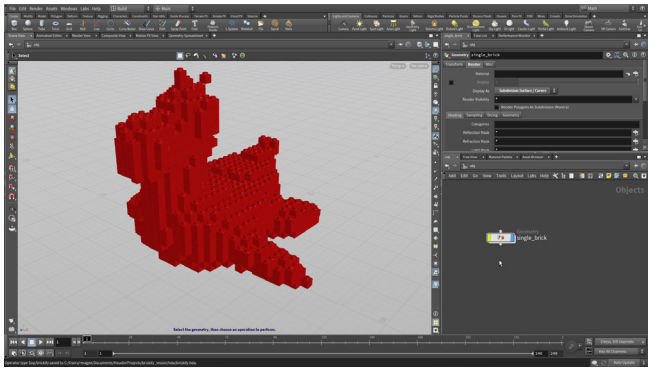
The asset is now using the *Mandril.pic* because it is the default.



11 Now press **Accept** to save the changes to your asset and close the Type Properties panel.

In the Network view, **press u** to go back up one level. With the *brickify_asset* node selected, choose **Assets > Lock Asset > Brickify** from the main menu. Press **Save Changes i**f prompted.

If you **double-click** on the *brickify_asset* node to dive down, you can see that the network is greyed out and these nodes are protected. You can only manipulate this asset using its parameters. You will have to unlock the asset to make changes to its inner workings.



12 Go to the Object level. Now **Save** your scene file to preserve the work you have done so far.

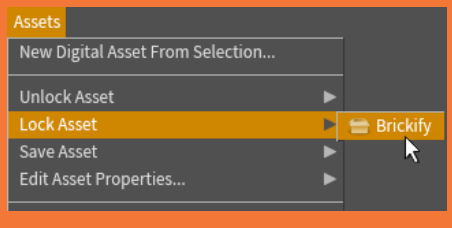
You now have the scene file and the .hda file which is being referenced into your scene to create the asset. You can use this library to create other instances of the asset in this scene or to add the asset to another scene.

In the next section, you will test out your asset and find out if it is working properly. It is always good to have one working asset and one for testing to make sure it is doing what you want it to do.

LOCKING AND UNLOCKING ASSETS

You can lock and unlock selected assets using the **Assets** menu. When the asset is **locked**, it references the HDA file to determine how the asset behaves. If the asset is **unlocked** the active definition is in your scene file. When you lock it you will be prompted to save if there are changes.

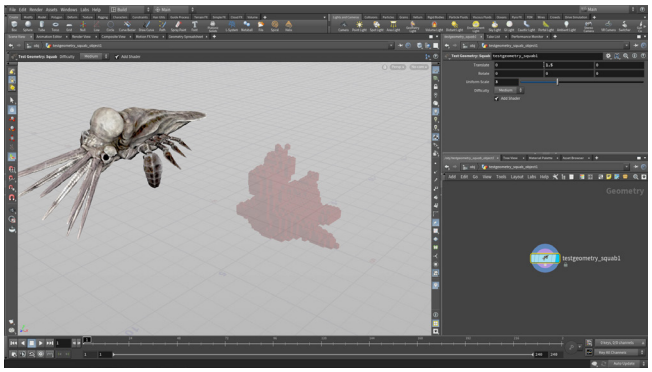
If you **RMB-click** on the asset node you can **Allow Editing of Contents** to unlock the asset or **Match Current Definition** to lock the asset but be careful because there is no prompt to save and changes might be lost.



PART SIX

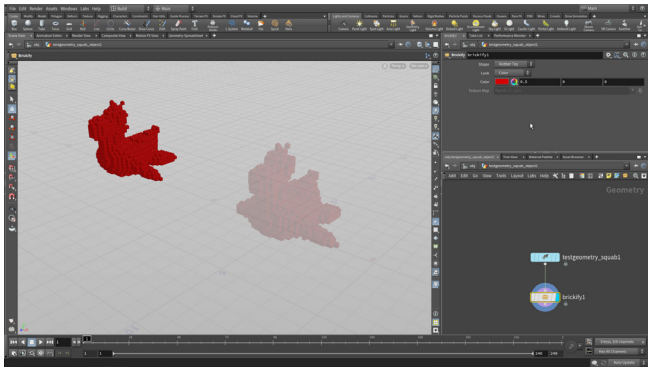
Test the Digital Asset

A Digital Asset can be instantiated more than once in a single scene file. You are going to use this asset on a different piece of geometry to test how it works. It is always good to have a test version available so that changes set to the first asset can be quickly verified. The asset can also be used in other scene files once you have it working properly.



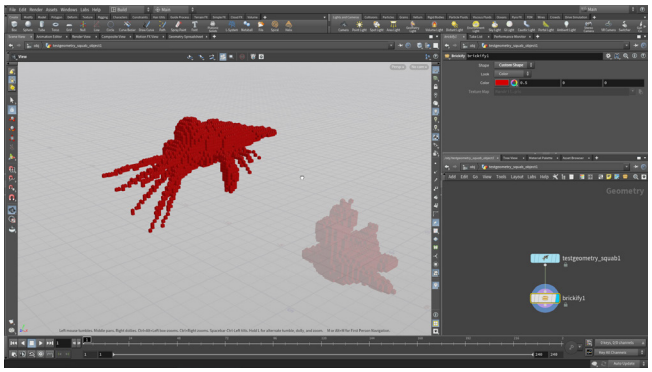
01 Use the **tab** key to get the **Squab Test Geometry**. Press **Enter** to place it at the origin then use the handle to move it to the side away from the rubber toy.

Double-click on the new object node to dive down to the geometry level. Select the node and set **Scale** to **3** and **Translate Y** to **1.5**. This will make the Squab a bit bigger than the rubber toy.



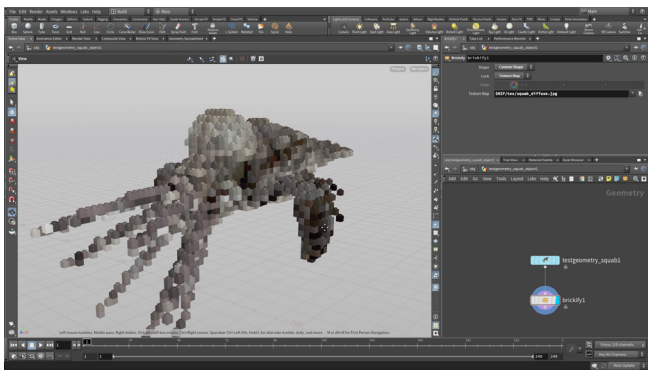
02 **RMB-click** on the output of the test geometry node and start typing **brickify...** then select the **brickify** asset from the menu. This places the asset into this new network.

Set its **display flag** and you will see another Rubber Toy colored **Red**. This is because these are the defaults for this asset.



03 On the *brickify* asset node, set the **Shape** parameter to **Custom Shape** and how the squab is being brickified. The new shape is running through the node network inside the asset to create a unique result. The shape will be lifted above the ground some more because of the **Match Size** node inside the asset.

This is how digital assets become tools that you can put into your pipeline to package up multiple actions into a single node. This is an approach that speeds up your workflow and helps you achieve more consistent results.



04 Select the *testgeometry_squab* node. From the **Asset** menu, choose **Edit Asset Properties > Squab**. In the **Properties** window, click on the **Extra Files** tab and select *squab_diffuse.jpg*. Click the **Save as File** button and save it into the *tex* folder. The texture was stored in the digital asset so that it could be shared along with the asset.

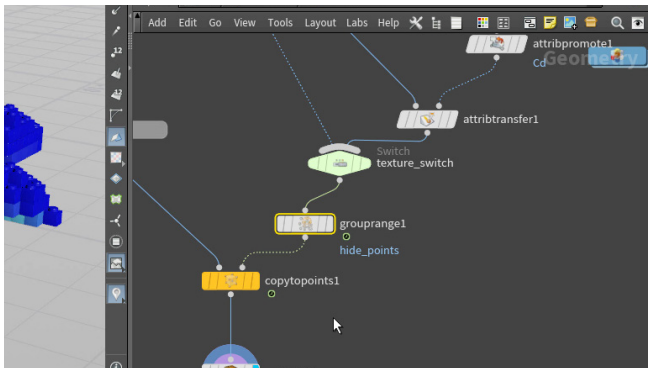
Now use this texture on disk to add color to the bricks using the *brickify* node's **Texture Map** parameter.

When you are finished, go to the object level and **name** this object *squab* and the other one *rubbertoy*. **Save** your work.

PART SEVEN

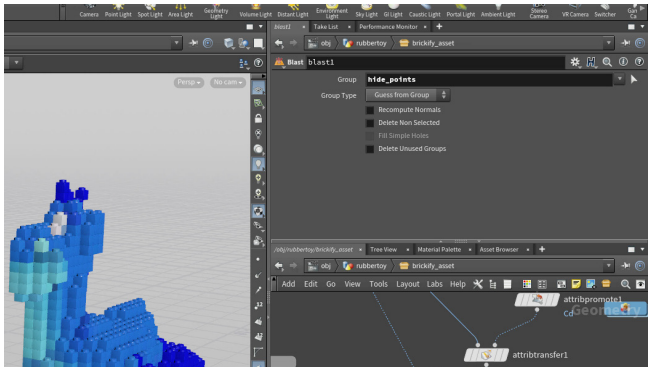
Animating the Bricks

It is possible to continue adding features to the asset and you will create an automating build-up animation for the bricks. This will involve adding more nodes to our network to make sure the asset has this new functionality. Once the results are saved into the .hda file, the features will be available for use by anyone using this asset in their work.



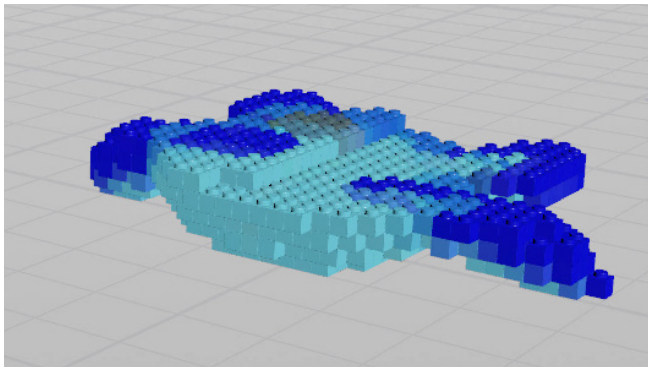
01 Hide the *Squab* object and dive into the *rubbertoy* object. Select *brickify_asset* and choose **Assets > Unlock Asset > Brickify**. **Double-click** on *brickify_asset* node then **RMB-click** on the output of *texture_switch* and select **Group by Range**. Place the node and set its **Display flag** then set the following parameters:

- **Group Name** to *hide_points*
- **Group Type** to **Points**
- **Range Type** to **Start and Length**
- **Length** to *($F-1)*20*
- Under **Range Filter**, leave **Select** to **1** and **Of** to **1**



02 **RMB-click** on the output of the *grouprange* node and select **Polygon > Blast.** Place this node down then using the arrow next to **Group**, select the *hide_points* group. Now turn on **Delete Non Selected** to delete points outside the group. Set the **Display flag** on the *blast* node.

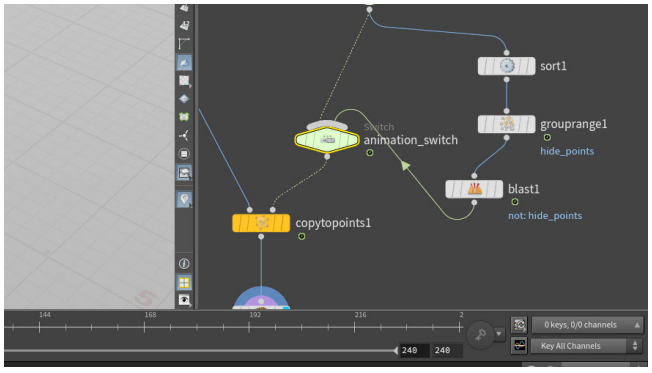
Press play to watch as the points grow with every frame. Now set the **Display flag** back to the *material* node at the end of the chain and watch the bricks grow over time.



03 Right now the bricks are coming in from one side instead of from the ground. This is because the points are appearing based on their point numbers. To control this, you need to reorder the points to create the look you want.

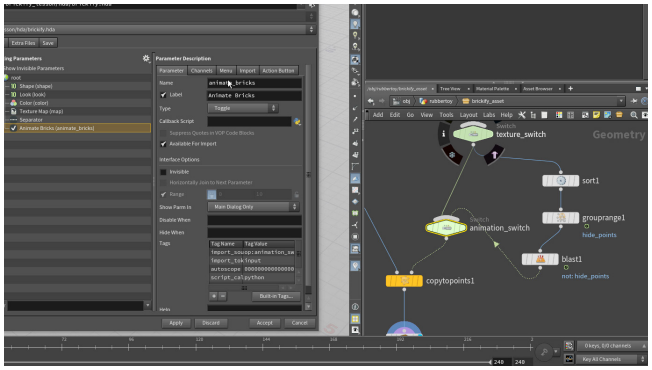
RMB-click on the output of the *texture_switch* node and type **Sort...** then select the **Sort tool.** Place this node down and change **Point Sort** to **Along Vector**. With this set to **0, 1, 0**, the points start at the bottom and go up.

Playback to see this result. Test out different vectors to see how it affects the animation.



04 To let you choose whether you want to animate the brickify effect, you can add another switch node. In the Network view, **Press tab** and start typing **Switch...** Choose **Switch** and place the node down. Rename it *animation_switch*.

Click on the output of the *texture_switch* node and feed it into the input of the *switch* node. Repeat for the *blast* node. This makes the original shape the first option and the animated effect the second option. Changing **Select Input** to **1** will reveal the animated bricks but for now keep it at **0**.



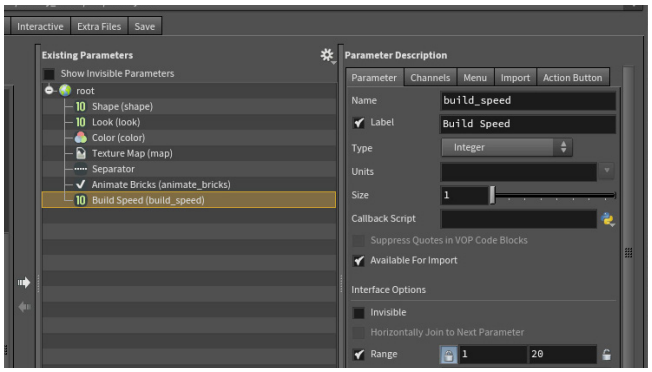
05

From the **Assets** menu, select **Edit Asset Properties > Brickify.** Go to the **Parameters** tab and drag a **separator** from the **Create Parameters** section to the bottom of the list.

Next, drag the *animation_switch* node's **Select Input** from the Parameter pane to just under the new separator. Set its **Name** to *animate_bricks* and its **Label** to *Animate Bricks*. Next, change its **Type** to **Toggle** which limits to you an **on[0]/off[1]** setting.

In the **Parameter Description** section, click on the **Channels** tab and set the default value to **0 [off]**.

Click **Apply** to save changes.



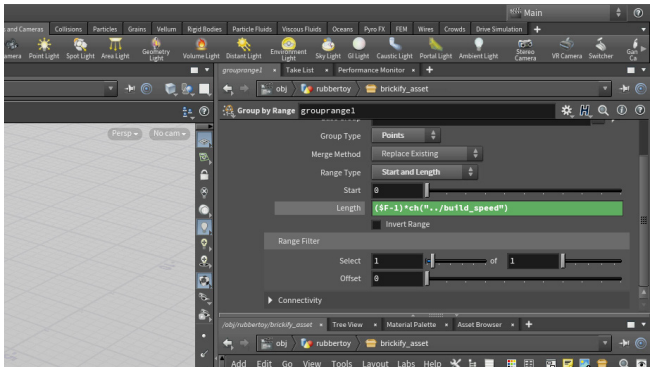
06

From the **Create Parameters** section in Type Properties, drag an **Integer** parameter under the *animate_bricks* parameter. Set its **Name** to *build_speed* and its **Label** to *Build Speed*.

Turn on the **Range** option then set the first value to **1** and the second value to **20**. Click on the **lock** next to 1 to make sure the number never gets smaller than 1.

In the **Parameter Description** section, click on the **Channels** tab and set the default value to **1**.

Press **Accept** to save and close the window.

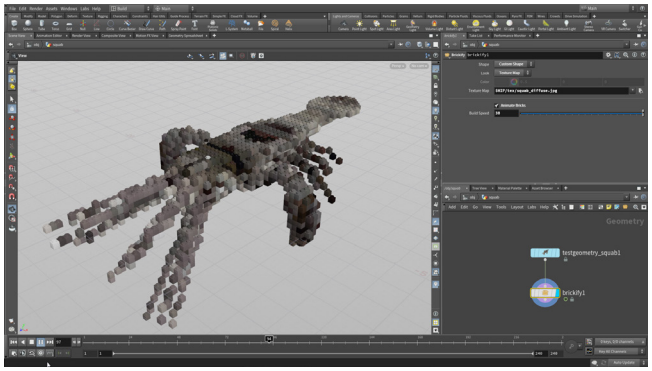


07

This parameter isn't attached to anything yet but you will now use it to drive the *grouprange* node's **Length** expression. Select the new *grouprange* node and change the **Length** expression to: `($F-1)*ch("../build_speed")`

At the end of the network there is an **output** node. This will ensure that you get the right output for your asset even if the **Display flag** is on another node in the network.

With the *brickify_asset* node selected, choose **Assets > Save Asset > Brickify** from the main menu. This lets you save this expression to the .hda file without re-opening the Type Properties window.



08

With the *brickify_asset* node selected, choose **Assets > Lock Asset > Brickify** from the main menu. You now have the brickify effect wrapped up into a custom tool which can be used on different shots by different artists.

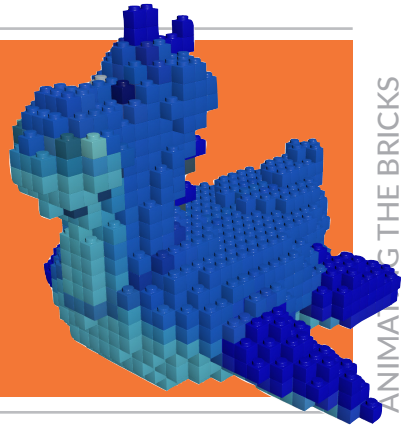
Go to the Squab network where the new features are available on the *brickify* node. Turn on the **Animate Bricks** toggle and set the **Build Speed** which might need to be set to around **30** because of the large number of bricks in this shape.

Playback to see the results.

CONCLUSION

You have now created a shareable tool which was built without writing scripts using a node-based workflow which is easily accessible to artists. By saving the HDA to disk, it becomes an asset on disk that can be referencing into multiple shots.

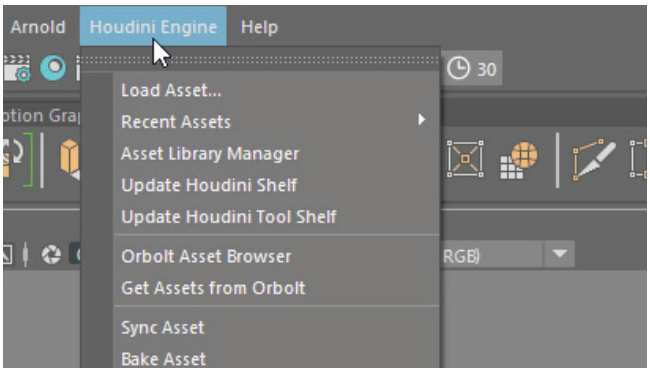
Houdini Digital Assets offer a powerful way for artists to share these kinds of tools in support of a studio-level production. These procedural assets make it easy to automate repetitive tasks and to stay focused on the creative needs of your project.



PART EIGHT

Loading HDAs into other Applications

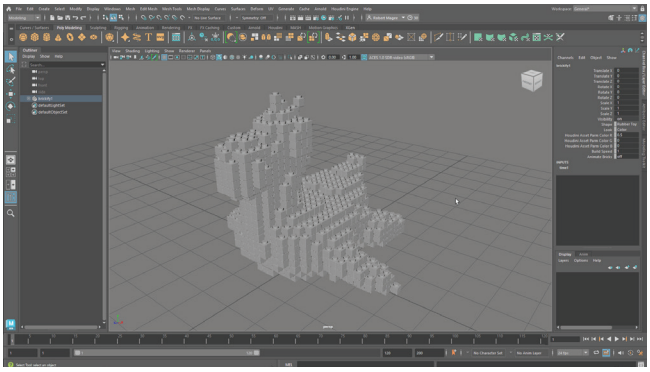
Once you have a Houdini Digital Asset [HDA] saved on disk, it is possible to load that asset into a host application using the Houdini Engine plug-ins. These let you share assets with colleagues who can load them directly into 3D apps such as Autodesk Maya or 3DS Max or into game editors such as Unity or Unreal Engine.



01 To use the Houdini Engine in a host application, you need to first install the plug-ins using the Houdini installer or Launcher. This will make the plug-ins available but you may need to take further steps to make the Houdini Engine available within your session.

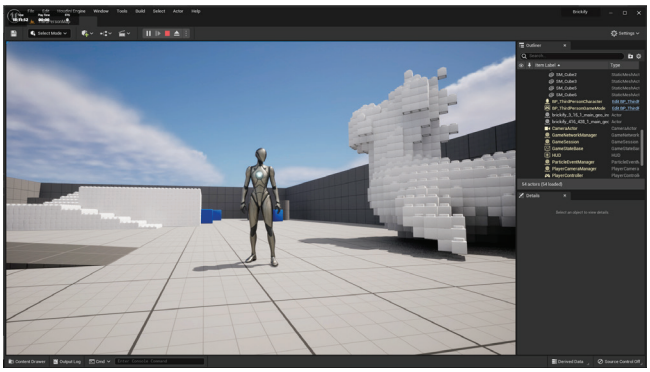
Visit the following page for details: **SideFX.com/engine**

Click on the Engine Plug-ins tab and then click on the desired plug-in for more information. Once installed, you will find a Houdini Engine menu in the host application. You can use this menu to load assets.



02 Once you have the plug-in installed, you can load the asset using the **Houdini Engine** menu. This will bring the *brickify* asset into the viewport while the asset parameters become available for manipulation.

You can also turn on animation for Maya or 3ds Max and play it the sequence using the timeline.



03 In Unreal, you can use the **Import** button to bring digital assets into your scene. You can also set **Shape** to **Custom Shape** and connect the asset to geometry within the host application and the brickification will be applied to that object.

WHAT HAPPENED TO THE BRICK COLORS?

You may notice that none of the brickified shapes are showing the brick colors within the various host applications. This is because the plug-ins don't always process information the same way as Houdini would. The point colors are still part of the asset but the host application is not receiving this information.

And while the animation works in Maya and 3ds Max, it would not make sense in Unity and Unreal Engine because Houdini assets can not become part of the runtime experience of a game therefore the built-in animation would be ignored. It is important to tailor your assets for the capabilities of the host application.

HOUDINI FOUNDATIONS

SMASHING WINE GLASS

In this lesson, you will smash a wine glass then slow down time to hold onto the big splash. This effect involves both an RBD simulation for the shattered glass and a fluid simulation for the wine. You will learn how to set up the dynamic network and output the simulations. Visual effect shots often involve a combination of different kinds of dynamic solvers and Houdini's dynamic network is designed to achieve a unified result with these different solvers.

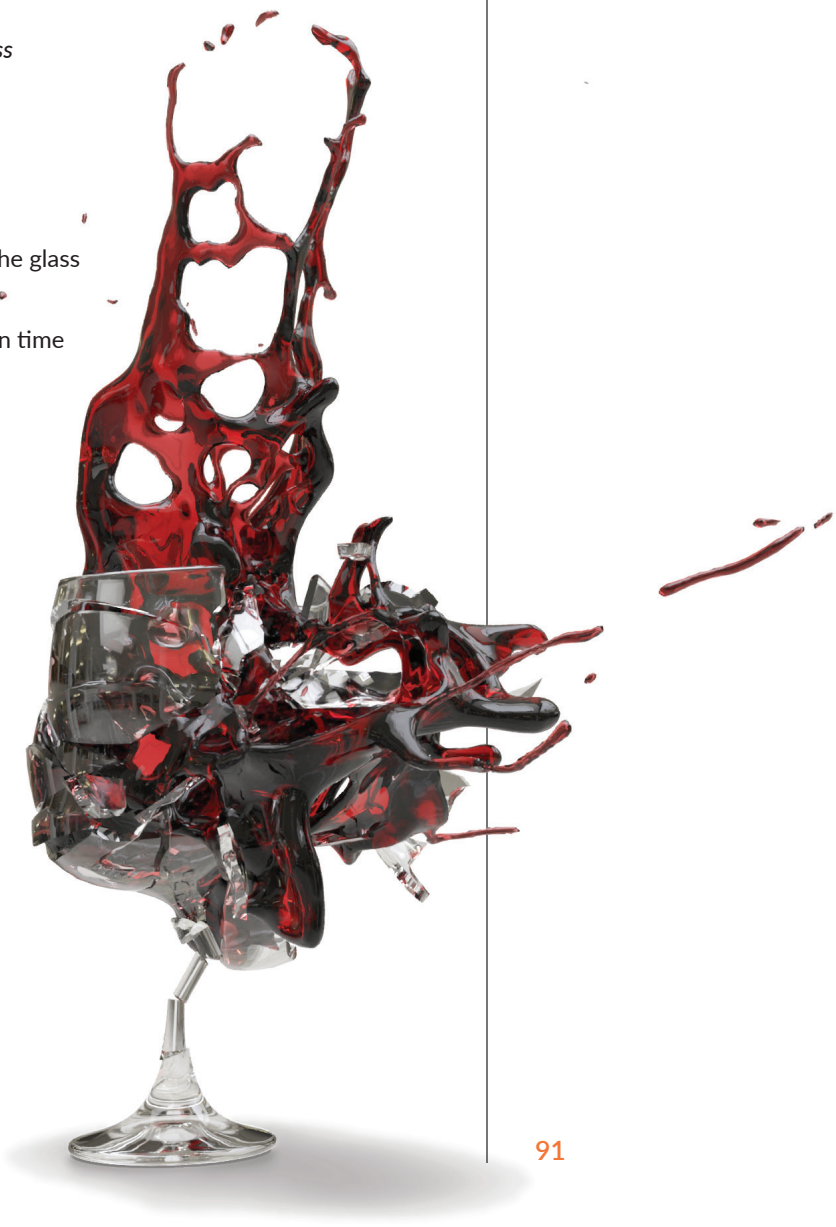
You will also use a retime node to slow down the simulation when it is at its most explosive then reverse time to return to the starting point. You will then move the simulations into Solaris/LOPS, set up lights and a camera, then render the shot using the Karma renderer.

LESSON GOAL

To simulate a bullet smashing through a wine glass causing the glass to break and the liquid to splash.

WHAT YOU WILL LEARN

- How to model the **Wine Glass, Bullet** and **Liquid** geometry
- How to **pre-fracture** the glass geometry using **Booleans**
- How to run a **Rigid Body Simulation** of the bullet smashing the glass
- How to run a **FLIP Fluid Simulation** of the liquid splashing
- How to **retime** the simulation to slow it down then go back in time
- How to export the results as **USD** for use in **Solaris/LOPS**
- How to set up **lights** and **materials** in **Solaris/LOPS**
- How to **render** the final shot with **Karma**



LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

Houdini Core	✘
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 2 | Sept 2022

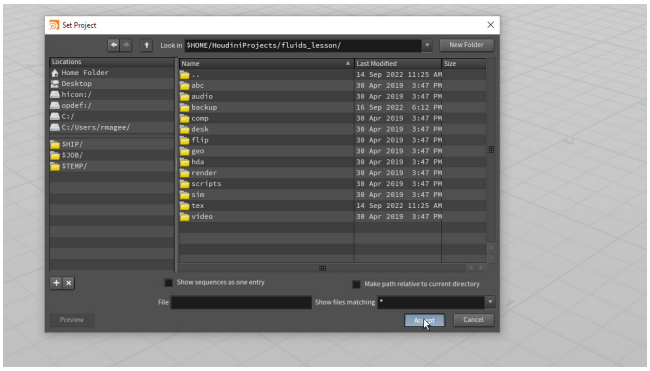
PART ONE

Model the Wine Glass

To start, you will draw a polygonal curve then revolve it to create the wine glass. Creasing will be used to sharpen some edges and then you will subdivide to create denser geometry for fracturing. You will then extract geometry from the wine glass to create a shape that you will use to simulate the fluid.

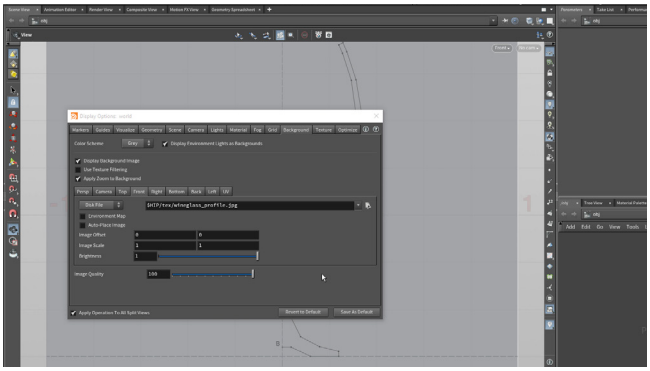
PROJECT FILES

Go to the **fluids tutorial** page on **SideFX.com**, where you got this document and download the *fluids_lesson_start* directory. Rename it *fluids_lesson* then put it into the **Houdini Projects** directory.



01 Select **File > Set Project.** Find the *fluids_lesson* directory (see instructions above) and press **Accept**. This makes the project directory and its sub folders the focus for all the files associated with this shot.

Select **File > Save As...** You should be looking into the new *fluids_lesson* directory. Set the file name to *wineglass_01.hip* and click **Accept** to save. Now you will be able to access the reference images in the *Tex* folder.

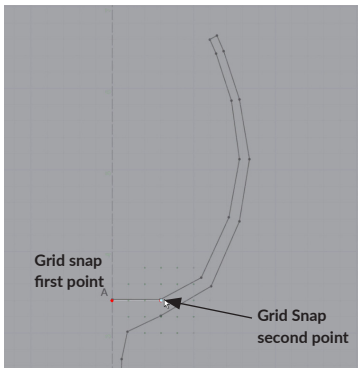


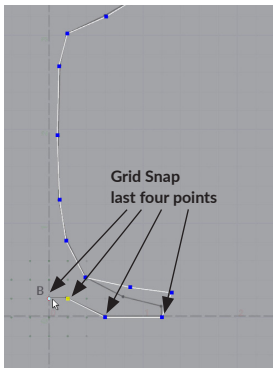
02 In the Scene view, **Press v** to bring up a radial menu and choose **Viewport Layout > Four Views** Move your cursor over the *Front* panel and **press spacebar-b** to expand it.

Hit **D** with the mouse in the viewport. Click on the **Background** tab and on the **Front** tab , use the file picker to navigate to *$HIP* then *tex>wineglass_profile.jpg*. Set the following:

- Make sure **Auto-Place** is turned **OFF**
- **Image Offset** to **0, 3**
- **Image Scale** to **5, 5**

Now if you dolly or pan the view, the background moves with it.

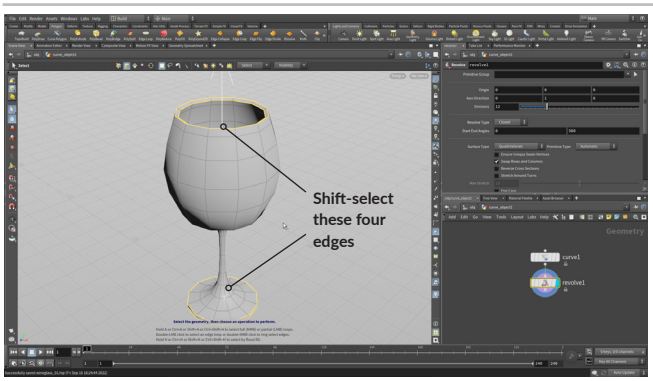




03 On the **Polygon** shelf, click on the **Curve Polygon** tool. This creates a **Curve** node with the **Primitive Type** set to **Polygon**. **Press x** and choose **Grid** to turn on grid snapping then click on **point A** and then to the second point which also sits on a grid point. Turn **Grid snapping** to **OFF** then keep tracing the image.

Turn **Grid snapping** back **ON** for the final four points at the base of the glass to make sure they are aligned. When you finish at point B, press **Enter** to complete the curve. Turn **Grid snapping** to **OFF**.

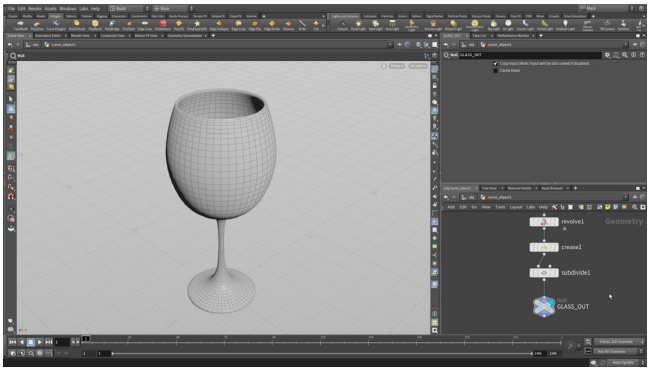
You can then click on the **Edit Mode** button in the Operation Controls bar to move any points that missed their mark.



04 Press **Spacebar-B** to go back to a four view layout then mouse over the perspective view and press **spacebar-b** again to expand it. Now you can see the curve in 3D.

Press s to get the **Select** tool and **press n** to select all the primitives on the curve. **Press c** and choose **Model > Curves > Revolve**. This will turn it into the wine glass model.

Press 3 to go to edge selection then **double click** on the top edge of the glass then press **Shift** and **double click** on the second top edge and the two edges of the base. Press **tab > Crease** then set **Crease** to **0.75**.



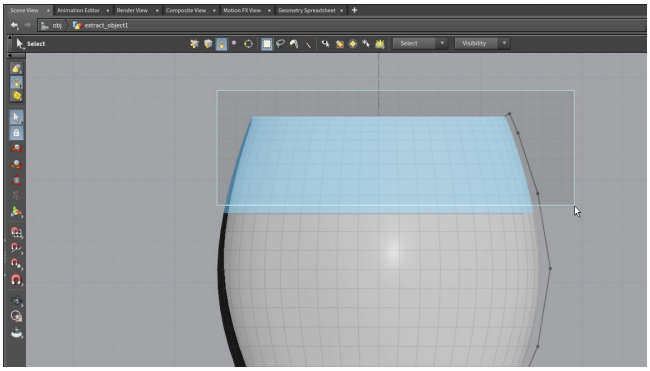
05

Press 4 to switch to primitive selection then **press n** to select all. Now **press tab > Subdivide**. Set **Depth** to **2**.

This will subdivide the model with the creased edges set up to be sharper than the other areas of the model. A higher **Crease** weight could make these edges completely sharp but for now a softer look works better.

In the Network editor, add a **null** node to the end of the chain and set its display flag. Rename this node to *GLASS_OUT*.

Go back up to the object level and rename the object to *wine_glass*.

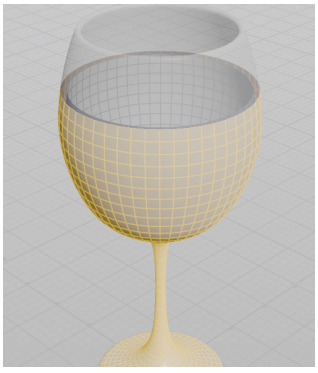


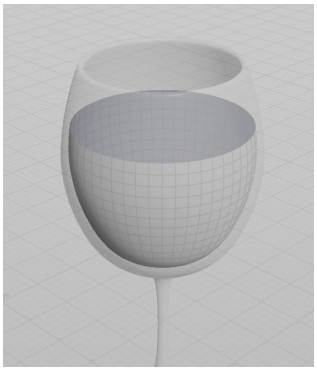
06

With the *wine_glass* node selected, **press n** to select all the primitives then go to the **Modify** shelf and click on the **Extract** tool. This will create an *objectmerge* of the wine glass geometry and place it into a new object.

From the Front view, **select** the faces at the top of the wine glass and then press **delete**. This adds a *Blast* node into the network that removes the faces.

Note: You can still see a ghosted version of the original wine glass because the Scene View is set to **Ghost Other Objects**. This setting is useful to add context to your work.

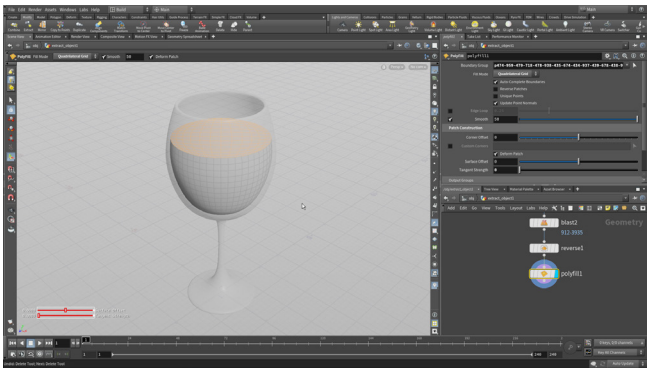




07

Now go back to a Perspective view and **double-click** on the base of the wine glass and press **delete**. This adds a second blast node. Now you are left with interior faces where you want the liquid to be. The faces of the wine geometry will appear dark on the outside. This means that they are the back sides of the primitives.

Press n to select all the primitives. **Press tab > reverse** to reverse the normals so that they are facing out. The darker side of each primitive will now be on the inside of the shape.



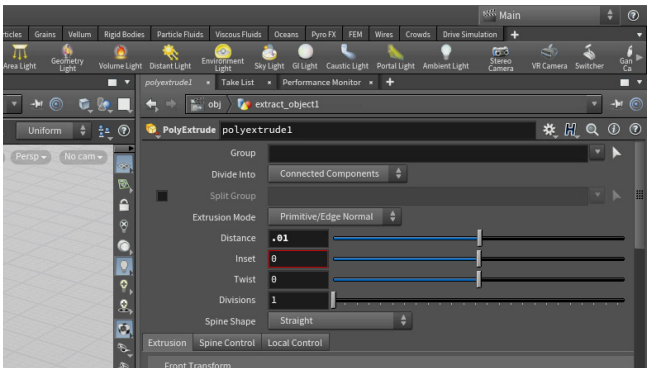
08

Press 3 to change to edge selection then double-click on the edge of the shape to select the open edge. **Press tab** and start typing **polyfill.** With **Polyfill** highlighted, press **Enter**.

Set the following:

- **Fill mode** to **Quadrilateral Grid**
- **Tangent Strength** to **0**.

This creates a closed shape that will become the source for the FLIP Fluid later in the lesson.



09

Press 4 to change to primitive selection then **press n** select all of the primitives. **Press c** and choose **Model > Polygons > PolyExtrude**. Set **Distance** to **0.01** to create an overlap with the wineglass to help the fluid render properly.

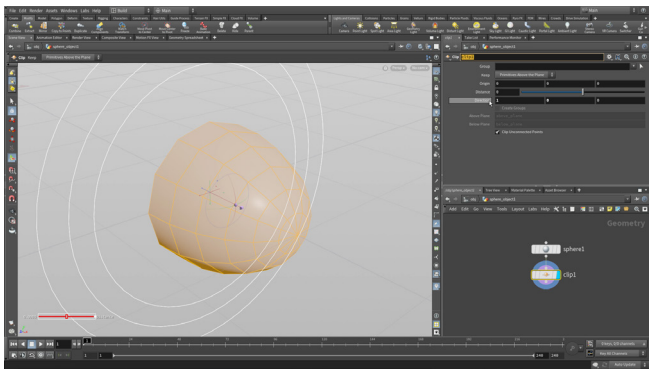
In the Network editor, add a **null** node to the end of the chain and set its **display flag**. Rename this node to *FLUID_OUT*.

Go up to the object level, rename this object to *wine*.

PART TWO

Model the Bullet

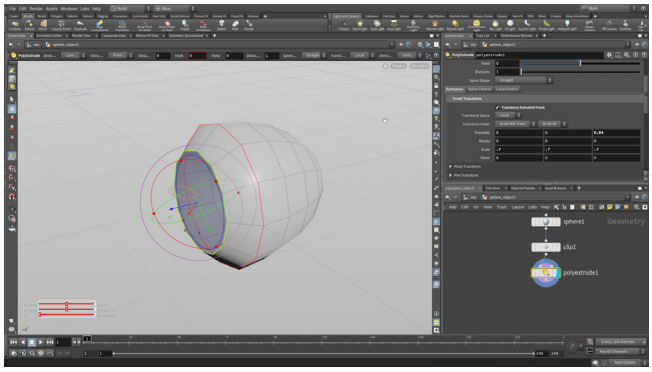
To create the bullet geometry, you will start with a primitive sphere and slice it in half. Next you will polyextrude the open end then use polyfill to close the shape using quad topology. You will then subdivide to define the final shape. This object will be moving very quickly therefore lots of detail isn't required.



01 Go to the object level and in the Network view, turn off the **Display Flags** for the *wine* and *wine_glass* objects. In the Scene View, **press c** and choose **Create > Geometry > Sphere**. Press **Enter** to place it at the origin then dive inside and set the following:

- **Orientation** to **X axis**
- **Radius** to **0.2, 0.125, 0.125**
- **Columns** to **12**

Press **tab > clip** in the Scene view then **press n** to select all the primitives and press **Enter** and set the **Direction** to **1, 0, 0**.

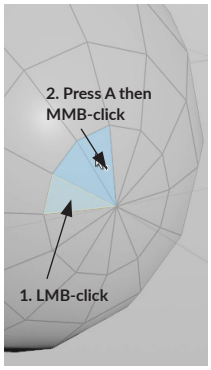


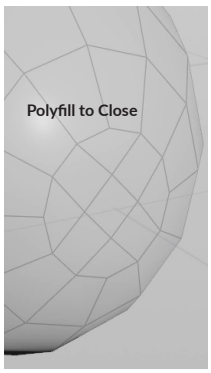
02 Click on the **Select** tool then **press 3** to turn on edge selection and **double click** on the open end of the sphere. **Press c** and choose **Model > Polygons > PolyExtrude**.

In the **Extrusion** tab, set the following:

- **Transform Extruded Front** to **ON**
- **Translate Z** to **0.04**
- **Scale** to **0.7, 0.7, 0.7**

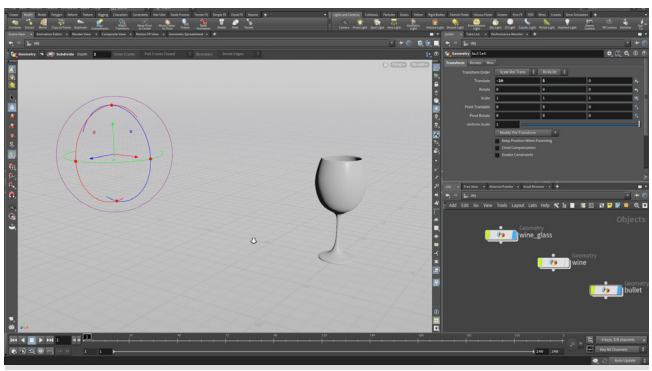
This adds a little extra geometry before you remove the triangles at the front of the bullet then close the shape.





03 Tumble around and **press s** to get the select tool and **4** to get **face/primitive** selection. Select one of the triangles at the tip of the bullet then **press and hold** the **a key** and then **middle click** two triangles over to select all the triangular faces. Press the **Delete** key to remove them. This adds a *blast* node to the network.

In the Network View, press **tab > polyfill** and place the node after the *blast*. Set **Fill mode** to **Quadrilateral Grid** and **Corner Offset** to **1** then turn on its **display flag**. This will close both ends of the bullet with good quad topology.



04 Now add a **Subdivide** node at the end, set **Depth** to **2** and then set its **display flag**.

In the Network editor, add a **null** node to the end of the chain and set its **display flag**. Rename this node to *BULLET_OUT*.

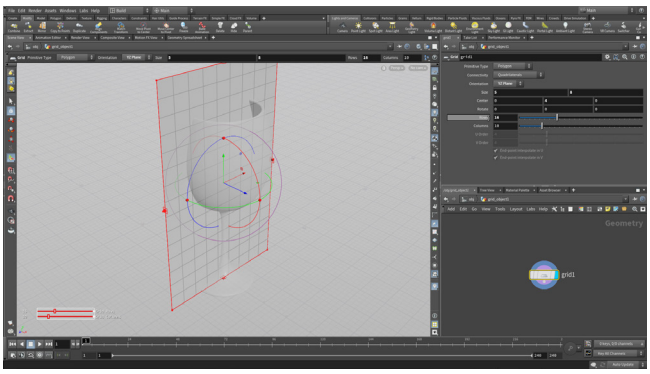
Go back to the object level and rename this object *bullet*. Turn on the **display flag** for the *wine_glass*. **Translate** the bullet to about **-20** in **X** and **5** in **Y**. You may want to go back to the *Front* orthographic view to make sure it is aimed at the desired impact point on the wine glass.

Save your work.

PART THREE

Fracture the Wine Glass

To define the cracks in the wine glass, you will create natural looking lines using the draw curve tool then extrude them into sheets of geometry. You will then agitate the surfaces using the mountain tool. This chaotic looking shape can then be merged into the wine glass object where you will set up a boolean operation that uses the sheets to shatter the glass.

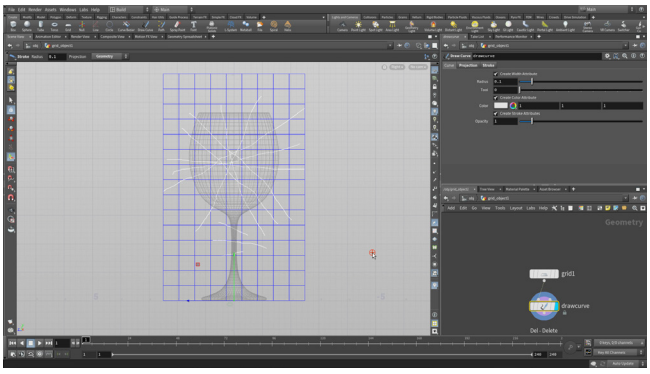


01

In the Scene View, go to the object level then **press c** and choose **Create > Geometry > Grid**. Press **Enter** to place it at the origin then **i** to dive inside and set the following:

- **Orientation** to **YZ Plane**
- **Center Y** to **4**
- **Size X** to **5**
- **Size Y** to **8**
- **Rows** to **16**

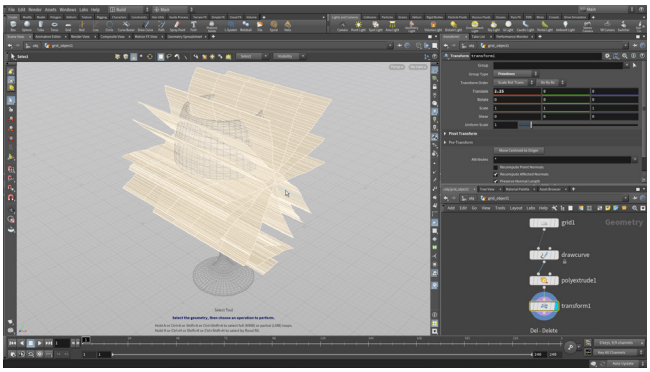
This will create a drawing surface for the **Draw Curve** tool that matches the shape of your wine glass.



02

Go to a *Right* view and **press v > Shading > Wireframe** to go into wireframe mode. You can see the bullet sitting behind the grid and the wine glass. **Press n** to select the whole grid then go to the **Create** shelf and click on the **Draw Curve** tool.

Draw curves over the wine glass that converge where the bullet hits the glass. Also add some curves across the stem of the glass since that will shatter as well. If at any point you draw a curve you don't like, press **Ctrl-Z** to undo it.



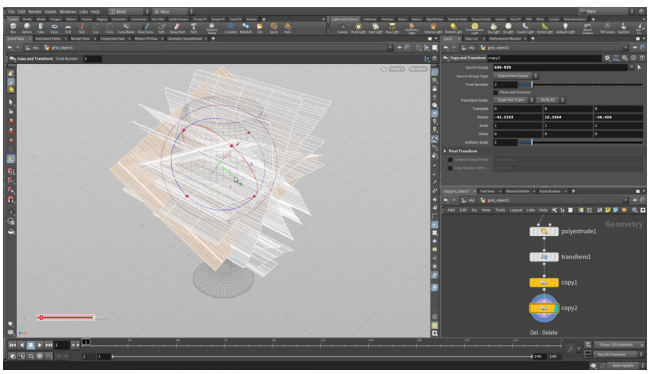
03

Go to the **Select** tool then **Press n** to **Select** all the curves then **press c** and choose **Model > Polygons > PolyExtrude**. Set **Divisions** to **4**.

In the **Extrusion** tab, set the following:

- **Transform Extruded Front** to **ON**
- **Transform Space** to **Global**
- **Translate X** to **-4.5**

Next, click away then select all the geometry and **press tab > Transform**. Set **Translate X** to **2.25** to center this geometry around the origin.



04

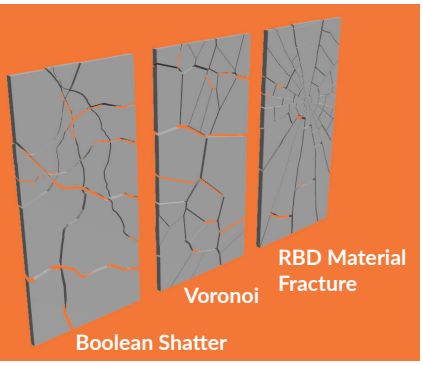
Press s to bring up select tool, then **press 4** to go to primitive selection. **Double click** on one of the sheets to select the whole thing. Press **tab > Duplicate** to create a copy. Use the **Rotate [r]** handle to reorient the duplicated sheet to be almost orthogonal to the others, making sure it cuts through the cup but avoids the stem. This will break the cup in the other direction to create more realistic slivers of glass.

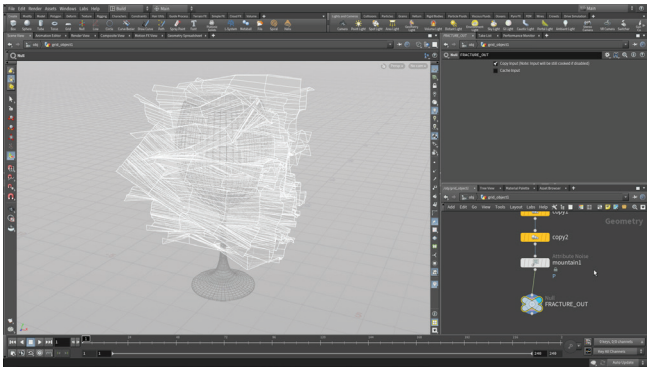
Repeat with another sheet to create another surface for cutting at a different angle.

BOOLEAN SHATTER

The Boolean node is often used to create traditional booleans such as **Union, Intersect** and **Subtract**. While these work well with closed shapes, you can use the **Shatter** option to slice up the geometry with sheets.

Houdini has a **Voronoi Shatter** tool that can also be used but it will not give you the jagged look you need for broken glass. There is also an **RBD Material Fracture** node that can create glass-like fractures. These work best with flat surfaces which is why it wasn't used in this lesson for the wine glass.

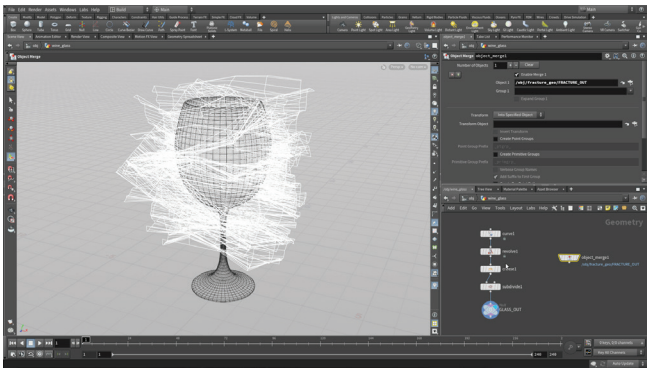




05 Select all the geometry and then press **tab > Mountain** to add some noise to the points on the different sheets. Set **Amplitude** to **0.75**. This will create more interesting cracks in the glass.

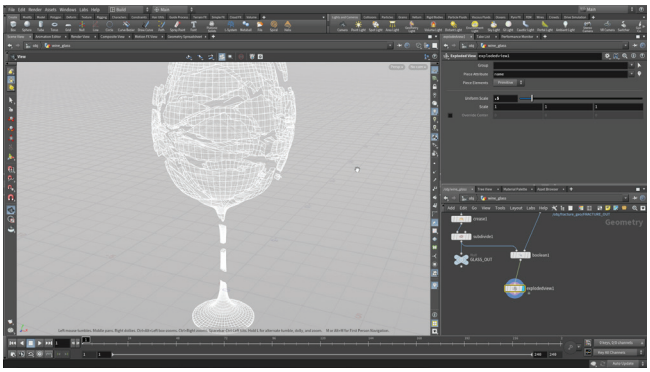
In the Network view, add a **Null** node to the end of the chain and rename it *FRACTURE_OUT*. Set the **display flag** on this node.

Go to the object level and name this node to *fracture_geo* then turn **OFF** its **display flag** to hide it.



06 Next dive into the *wine_glass* object. In the Network view, press **tab > object merge** and place the node down. Click on the node selector next to **Object 1** and navigate to **fracture_geo > FRACTURE_OUT** and select that node. Make sure that **Transform** is set to **Into Specified Object** .

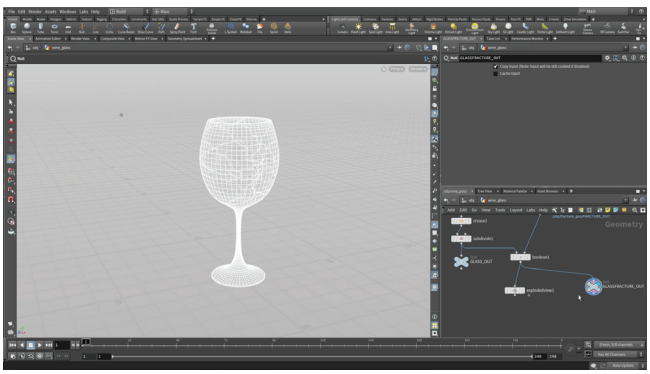
This brings the agitated sheets into the wine glass geometry network where you can use it to boolean shatter the glass.



07 In the Network view, press **tab > boolean** and click to place down the new node. Wire the *subdivide* node (not *GLASS_OUT*) into the first input and the *object_merge* into the second. Set its **Display flag** then set the following:

- **Set B: Treat As** to **Surface**
- **Operation** to **Shatter (Pieces of A)**

To visualize the cracks, add an **exploded view** node at the end of the chain. If you want to change how things work then you can go back to the *fracture_geo* object and edit the sheets. Those edits will update procedurally.



08 In the Network view, add a **Null** node that bypasses the *exploded_view* node and rename it *GLASSFRACTURE_OUT*. Set the **display flag** on this node.

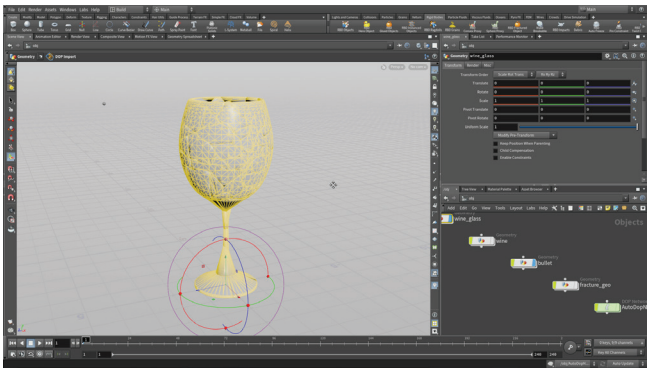
You now have two output nodes for this network. GLASS_OUT gives you the wine glass in its original form and GLASSFRACTURE_OUT gives you the shattered glass. You will use them both down the line to complete the shot.

Save your work.

PART FOUR

Set up the RBD Simulation

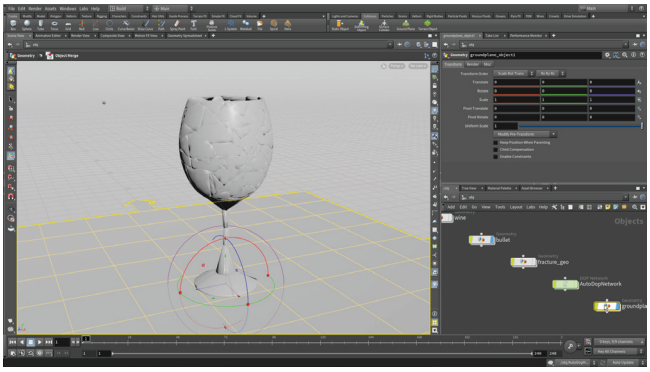
You are now going to create a rigid body simulation using shelf tools. This will add a DOP (Dynamic Operator) network that brings together geometry, forces and a solver node. In the wine glass geometry network, nodes will be added to prepare the geometry for simulation. You will use convex proxies so the Bullet RBD solver can handle the odd-shaped pieces of glass.



01

Go up the object level, select the *wine_glass* object then from the **Rigid Bodies** shelf, click on the **RBD Convex Proxy** tool. This will set up the initial dynamics network for you. It is called *AutoDopNetwork*. **Press v > Shading > Smooth Wire Shaded.**

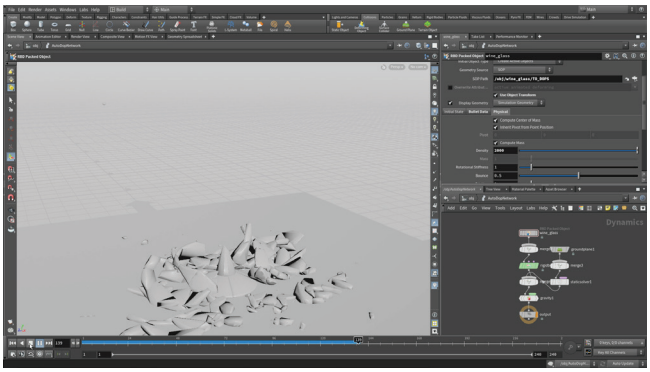
This tool is used to break down the parts of the wine glass into convex shapes that can be used to create complex collisions. These will appear rougher than the source geometry but after the simulation, you can go back to visualizing the cleaner topology.



02

Go to the **Collisions** tab and click on the **Ground Plane** shelf tool. This will create an infinite ground plane for your geometry to collide with.

You can turn **Off** the **Display flag** on the *groundplane_object* to hide it from the scene view. It will still function as a colliding surface in the simulation.



03

Dive into the *AutoDopNetwork* node. Select the *wine_glass* node and in the **Physical** tab set **Density** to **2000**, which is approximately the density of glass.

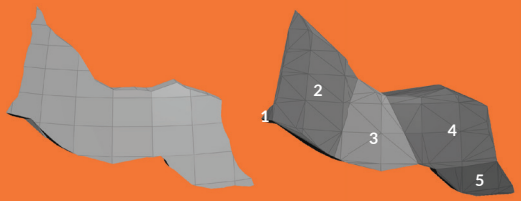
Press **Play** in the playbar to see what happens. The glass falls and hits the ground. Right now gravity is the only force acting on the pieces.

You could set up a **glue network** to hold the pieces together until impact but the bullet will be so fast that connecting the parts is not necessary.

CONVEX DECOMPOSITION

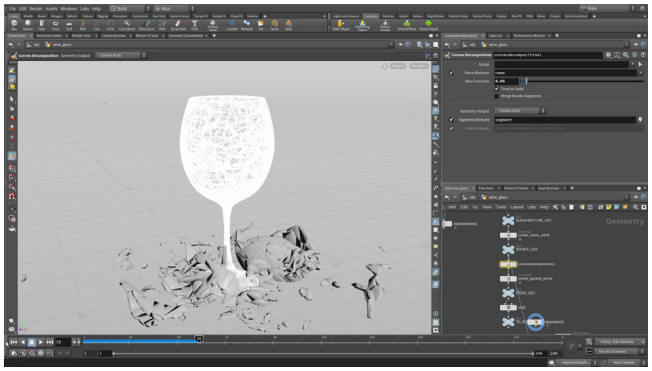
For RBD simulations, Houdini uses the Bullet solver which prefers convex shapes in order to maintain fast simulation speeds. **Convex Decomposition** lets you take shapes that are concave and break them down into convex shapes that are connected together. These are then simulated as one composite piece by the Bullet solver.

Since the wine glass fragments come in a variety of shapes, convex decomposition ensures accurate collisions for all pieces.



Original Geometry

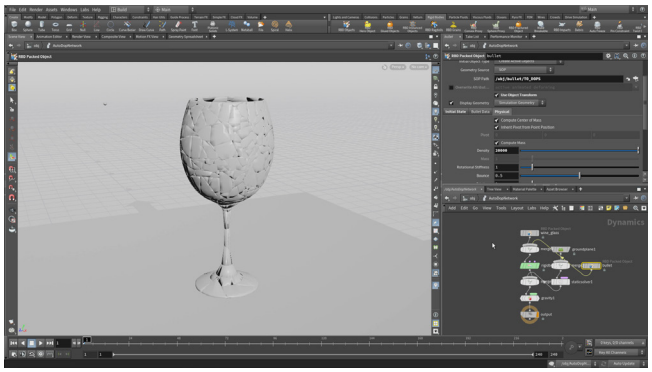
Convex Decomposition



04 Go up to object level and dive into the *wine_glass* object. A number of nodes have been added to create the proxy geometry and this chain ends with the *dopimport* node which is currently displayed.

On the *convexdecomposition* node, you can change **Max Concavity** to **0.05** to adjust your collision geometry.

Press **Play** to watch it simulate.

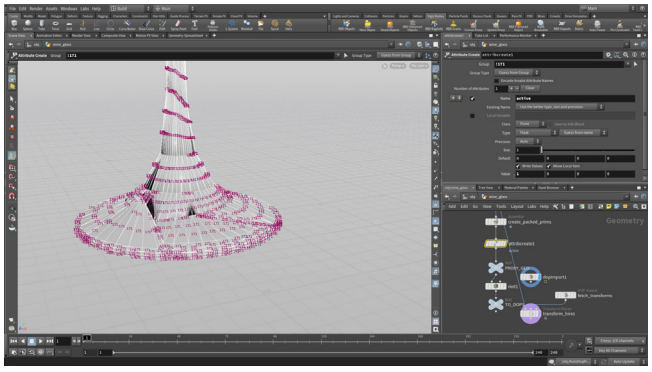


05 Go up to object level and select the *bullet* object and then go to the **Rigid Bodies** shelf and click on the **RBD Objects** tool. This will create a packed rbd object from the bullet and add it to the dynamics network.

Navigate into *AutoDopNetwork* and select the *bullet* node.

- In the **Initial state** tab, set **Velocity** to **400, 0, 0**.
- In the **Physical** tab set **Density** to **20000**

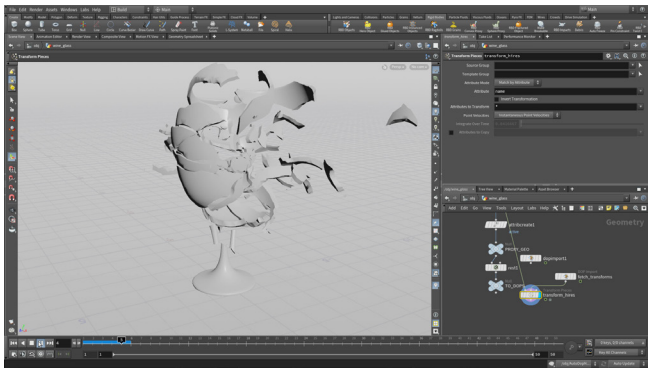
This is the density of lead and should work well for the bullet.



06 Before you simulate, you want to make sure that the base of the glass stays on the ground. Navigate into the *wine_glass* object and in the display bar, turn on **primitive** numbers. You can see that in this case the base packed primitive is 171 - yours will probably be different.

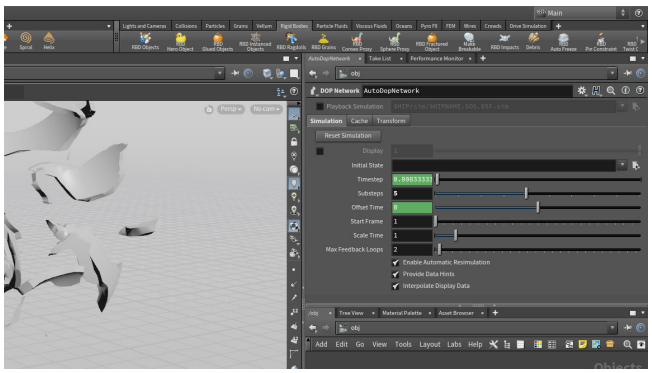
Add an **Attribute Create** node between the *create_packed_primitive* and *proxy_geo* nodes. Set the following:

- **Group** to **!171** (or whatever number your base is)
- **Name** to *active*
- **Value** to **1, 0, 0, 0**



07 Next, set the **display flag** on the *transform_hires* node and open up **Global Animation Options** and set **End** to **50.**

Now **play** back the sim. You can now see the source geometry being animated to match the proxies but the collision is as dramatic as it could be.



08 The Bullet sim defaults to **10 substeps** on the solver. This isn't enough to resolve collisions for the speed of the of the bullet. Go up to the object level and on the *AutoDopNet* node set **Substeps** to **5**. This will add substeps on top of the solver setting. This may take longer to simulate but will increase accuracy and generally make the simulation more active.

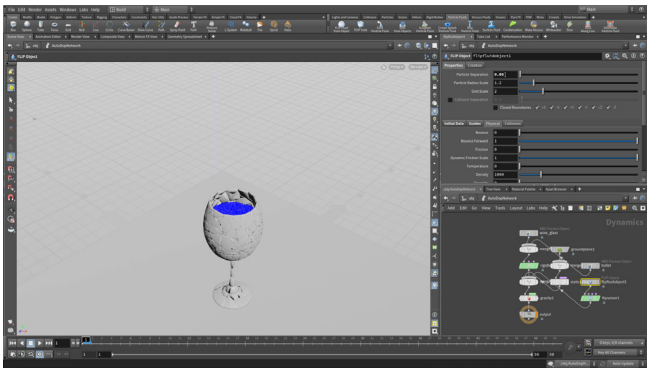
Play the simulation again to see that the collisions are looking much better. Go back and tweak settings if you want too change things. You can even go back and reposition some of the cracks.

Save your work.

PART FIVE

Add Fluids to the Simulation

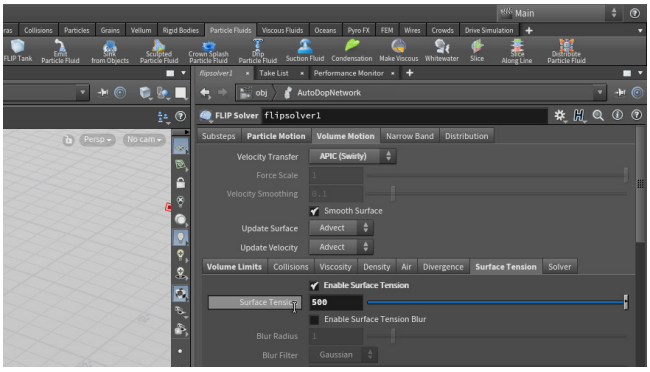
Now that the bullet is smashing the glass, it is time to convert the wine object into a fluid that will become part of the integrated simulation. This means that the RBD and Fluid simulations take place in the same DOP network and will work as one system. At first, the fluid will be represented by particles that can then be surfaced to visualize the fluid.



01

Go to **frame 1**. Select the *wine* object in the Network view and from the **Particle Fluids** shelf, click on the **FLIP Fluid from Object** tool.

This turns the fluid into a volume of fluid particles. In *AutoDopNet*, select the *flipfluidobject* node and set **Particle Separation** to **0.05**. This creates more particles which in turn adds more detail to the simulation.



02

Select the *flipsolver1* node and click on the **Particle Motion** tab, under **Behavior** set:

- **Add ID Attribute** to **ON**

Under **Reseeding** set:

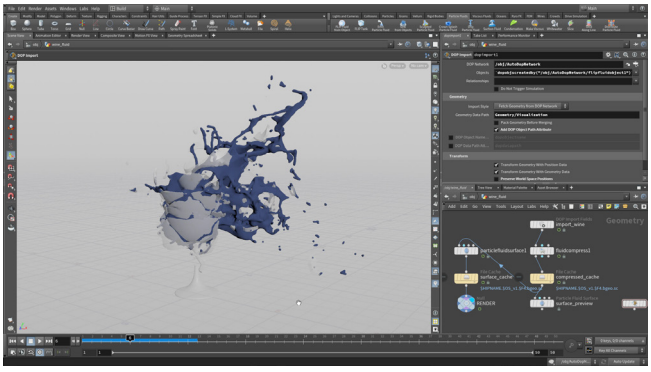
- **Reseed Particles** to **OFF**

In **Volume Motion**, set:

- **Velocity Transfer** to **(APIC) Swirly**

Under **Surface Tension**, set:

- **Enable Surface Tension** to **ON**
- **Surface Tension** to **500**



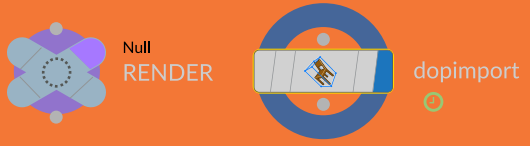
03

Press **Play** to run the simulation. With extra substeps it will take a little longer but will give you more accurate results. Simulate to just past **frame 10** and press **Escape** to stop the sim. Go to **frame 10** to preview the fluid so far.

To see it as a surface, go up to the object level and go into the *wine_fluid* object. Set the **display flag** on the *Render* null node to see the surfaced fluid. This will take longer to cook but will show you what the surface is going to look like at each frame.

DISPLAY AND RENDER FLAGS

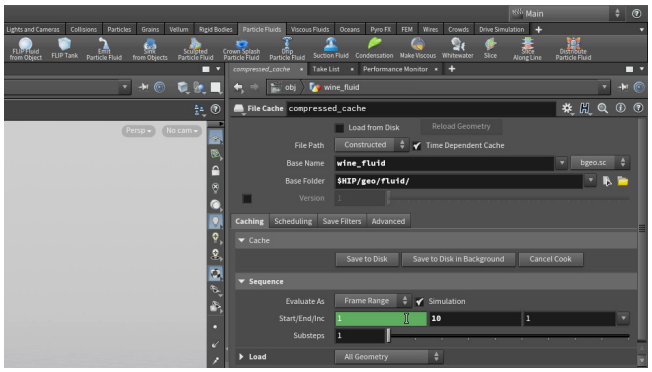
When the **wine_fluid** geometry network is first created, the **Display flag** is on the **dopimport** node which shows particles while the **Render flag** is on the null node which flows from a **particle surface** node. This set up is meant to give you fast performance in the viewport and if you render then you see the final surface. In this lesson, you will be caching out the surface therefore these nodes will not be used to render.



PART SIX

Cache and Retime the Simulations

For this shot, you only need to compute 10 frames of the simulation. You will save this to disk then use a retime node to create a longer shot where the fluid slows down then reverses in time. The retimed fluid particles will then be surfaced and output as a 50 frame sequence which will define the final shot for rendering. The wine glass and bullet will then be retimed to match.

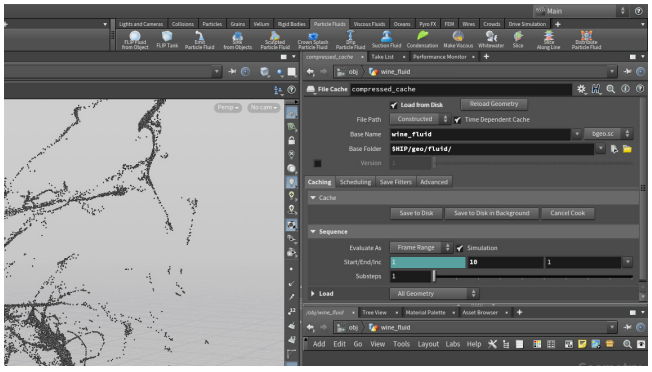


01 At the object level, **delete** the *wine_fluid_interior* node that the shelf tool created. Dive into the *wine_fluid* object. Delete all the nodes except the *import_wine*, *compressedcache* node and the *particlefluidsurface* nodes.

On *compressed_cache*, set the following:

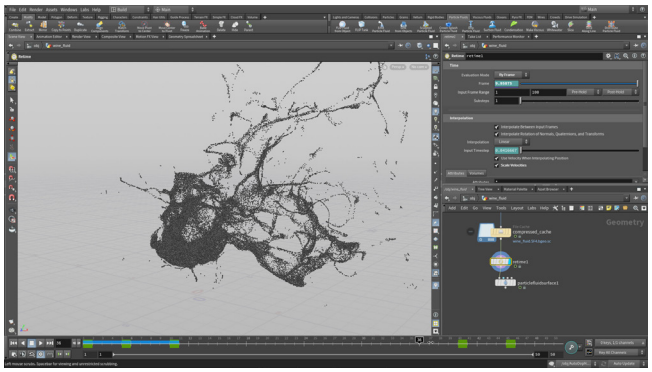
- **Base Name** to *wine_fluid*
- **Base Folder** to *$HIP/geo/fluid/*
- Turn **Off** the **Version** check box
- **End** to **10** (**RMB-click > Delete Channels** first)

Hit **Save to Disk.**



02 When it finishes, stay at the geometry level. From the **Visibility** menu in the top right of the Scene view, select **Hide Other Objects**.

Set **Load from Disk** to **On** and **Scrub** through the geo sequence. It will only play for 10 frames. You are now going to retime the sequence to stretch it out over 50 frames and slow down the effect.

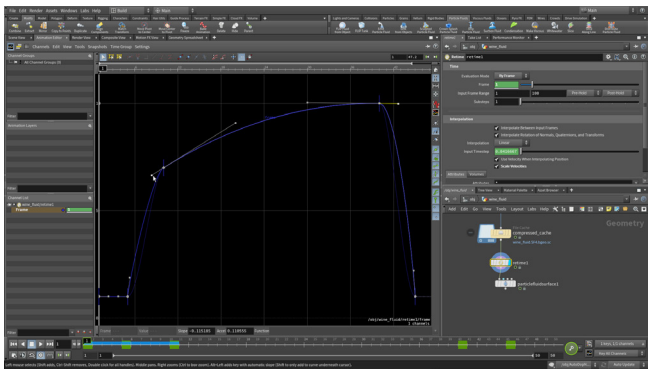


03 Add a **Retime** node after the *compressed_cache* node and set its **Display flag**. Set the following:

- **Evaluation Mode** to **By Frame**.
- Turn **ON** the **Scale Velocities** option

RMB-click on the **Frame** field and choose **Delete channels**.

- At **frame 1**: set **Frame** to **1** and **Alt-click** to keyframe
- At **frame 5**: set **Frame** to **1** and keyframe
- At **frame 10**: set **Frame** to **7** and keyframe
- At **frame 40**: set **Frame** to **10** and keyframe
- At **frame 45**: set **Frame** to **1** and keyframe



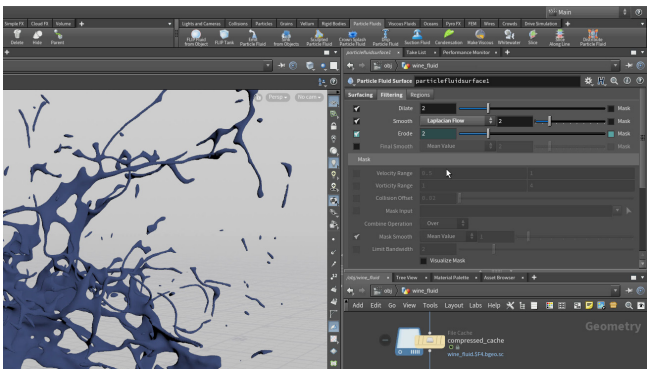
04 Click on the **Animation Editor** pane to see the animation curve you just created. **Press h** over the graph to home the view and then **RMB-click-drag** to zoom out a bit.

Click and drag on the curve handles to get a shape similar to what you see here. If you need to break tangents on a curve, select the key and **press T** to untie the tangents. Then select and drag on each end individually. The goal here is to have the liquid splash out quickly then slow down until is freezes for a short time then speed out into a fast reconstruction back to its original shape.

RETIME

The most interesting part of the smashing wine glass is the first 10 seconds. To emphasize this part of the simulation, you are going to save out the fluid particles for the 10 frames then uses the **retime** node to stretch out the sequence in a sort of "bullet-time" effect that snaps back to the original wine glass by reversing time.

Once you have this set up for the fluid, you can surface the points and save out a longer sequence. The same retime node can then be copied and pasted to be used on the smashing glass and the bullet.



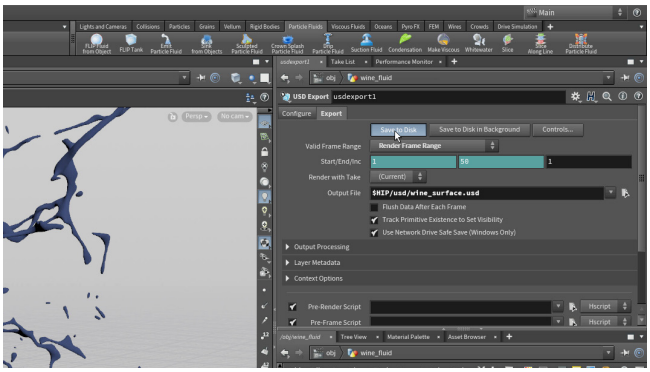
05

Make sure that the *retime* node is wired into the *particlefluidsurface* node then set it's display flag. This will give you your final fluid based on the retiming. Select the *particlefluidsurface* node and keep:

- **Method** set to **Average Position**
- Set **Union Compressed Fluid Surface** to **Off**.

In the **Filtering** tab turn:

- **Dilate** to **ON** and set it to **2**.
- **Smooth** to **ON** and set it to **Laplacian Flow**

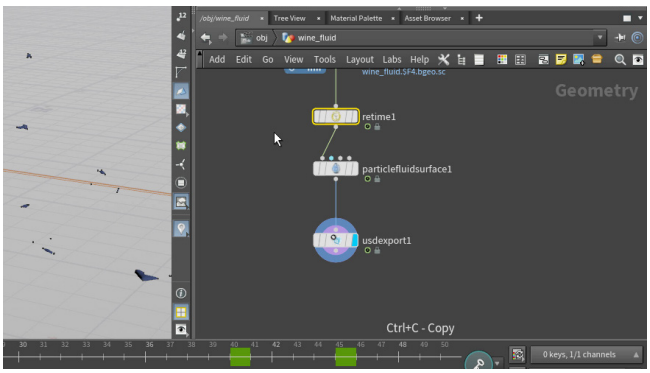


06

Add a **USD Export** node to the end of the chain. **Rename** it *wine_surface*. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/wine_surface.usd*

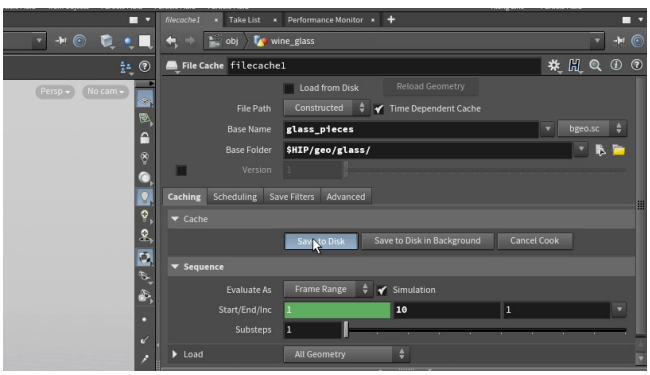
Hit **Save to Disk.**



07

Select the *retime* node and press **Ctrl-c** to copy it. You will paste this in another network to retime the shattering wine glass. This will ensure that the keyframes match in both of the networks.

Save your work.



08

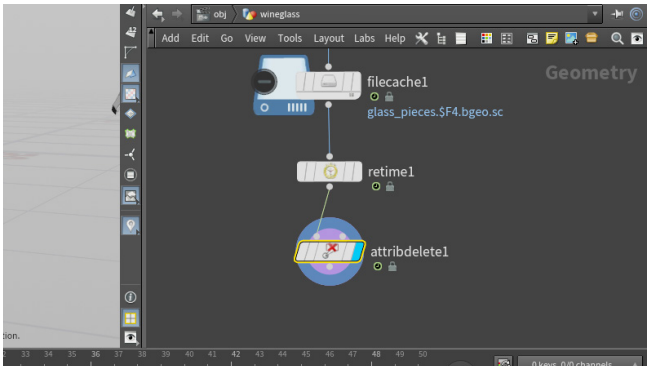
Go to frame 1 and navigate into the *wine_glass* network. Beneath the *transform_hires* node, wire in a **File Cache** node and set the following:

- **Base Name** to *glass_pieces*
- **Base Folder** to *$HIP/geo/*
- Turn **Off** the **Version** check box
- **End** to **10** (**RMB-click > Delete Channels** first)

Hit **Save to Disk**. Set **Load from Disk** to **On** and **Scrub** through frames 1 to 10 to make sure that it looks correct.

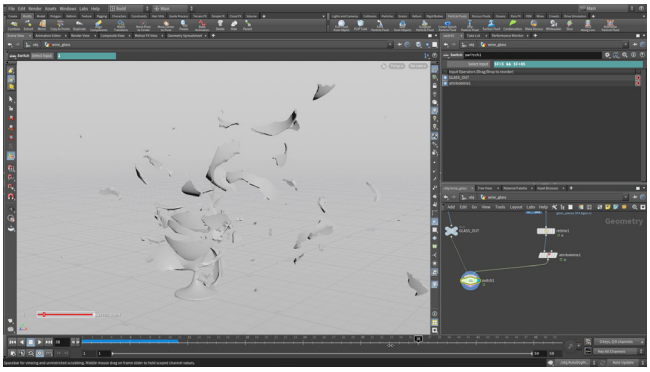
USD and SOLARIS

To support the look dev stage of this project, you are caching out the **fluid**, the **wine glass** and the **bullet** to USD. By doing this, you can focus on rendering without worrying about simulations being recalculated. Here you will display the caches in the same scene file as the simulations but another option would be to start a new scene file and import the caches into that file. This would let you focus on lighting and rendering your shot but would make it harder to go back and tweak the sim.



09 **Press Ctrl-v** to paste in the *retime* node and place it after the *filecache* node and set its **Display flag**. Now you have the same timing on the glass that you have on the fluid. You can scrub through the timeline to watch the timing of the pieces.

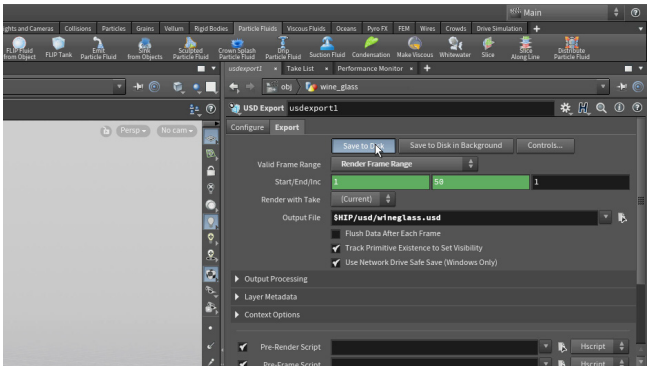
After the *retime* node, add an **Attribute Delete**. From the arrow next to **Primitive Attributes**, select *name* to remove this attribute from the geometry.



10 Add a **switch** node into the network. Feed the *GLASS_OUT* into it first and *attribdelete* node into it second.

Set **Select Input** to *$F>5 && $F<45*. Scrub in the timeline to see how this works. This expression makes the switch from the unbroken glass to the broken glass at frame 5 and then back at frame 45.

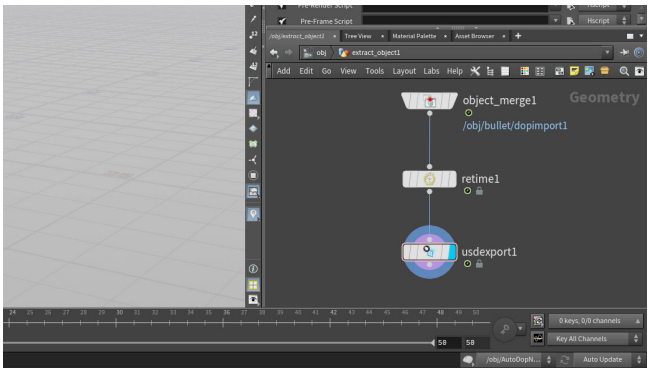
You are putting these shapes together so that the glass is unbroken before and after impact in the wineglass USD file.



11 Add a **USD Export** node to the end of the chain. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/wineglass.usd*

Hit **Save to Disk.**



12 Go to **frame 1** and select the *bullet* object. From the **Modify** menu, select **Extract**. This gives us the world space position of the geometry. Dive into *extract_object* and, **Paste** the *retime* node beneath the *object_merge* node then connect them.

Add a **USD Export** node to the end of the chain. **Rename** it *bullet*. Set the following

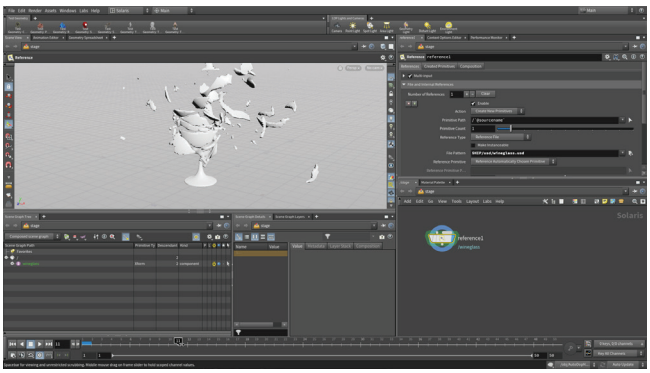
- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/bullet.usd*

Hit **Save to Disk.**

PART SEVEN

Set up and Render the Shot

To render the shot, you will reference the USD files into the Solaris Stage then add a backdrop. Solaris is the context of Houdini that uses LOP nodes to set up the USD Scene Graph. Next, you will add and position a camera and then an environment light. The Karma renderer will then be invoked in the viewport to create a preview render of the shot.

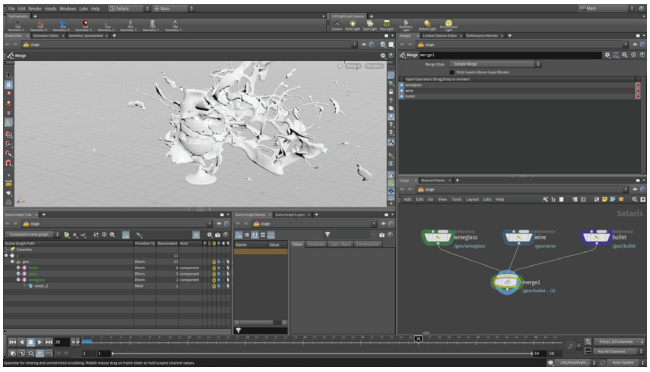


01

Change the desktop to **Solaris**. Choose **Stage** from the path bar.

In the Network View press **tab > Reference** then click to add a **Reference** node. Next to **Reference File**, click on the **File Chooser** and find the *wineglass.usd* file. Rename the node to *wineglass*. Set the **Primitive Path** to */geo/$OS* - this will use the node name and place it into a group called *geo*.

In the Scene View, use your view tools such as **spacebar-h** for homing the view to get a better look at the wine glass.



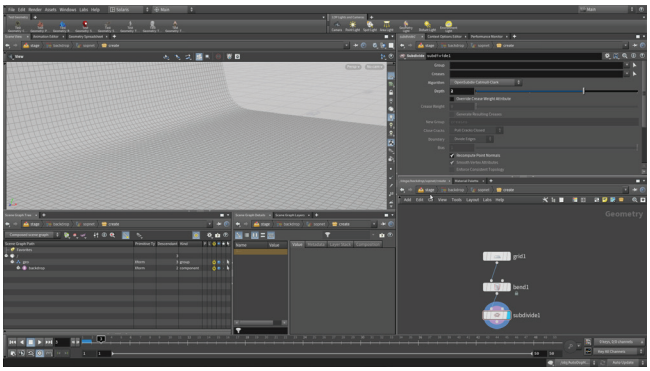
02

Alt-drag on this node to make a copy and again to make a second copy. For the first copy, click on the **File Chooser** and find the *wine_surface.usd* file. Rename the node to *wine*.

Repeat to make another copy, find the *bullet.usd* file. Rename the node to *bullet*.

Add a **Merge** node to the network and wire all three reference nodes into it. Set its **display flag** then scrub to see the results.

In the **Scene Graph**, you can see a *geo* entry and underneath that you will find the three referenced USD files.



03

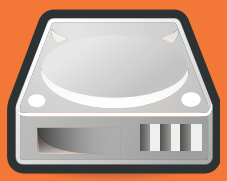
In the Network view, press **tab** and type out **Grid**. Click to place down the node, rename it *backdrop* and wire it into the *merge* node. Set **Import Path Prefix** to */geo/$OS*. **Double-click** on the *backdrop* node to dive down to the geometry level.

Select the *Grid* node and set the **Size** to **200, 200** and **Rows** and **Columns** to **20**. **RMB-click** on the *grid* node's output and type **Bend**. Click to place a bend node and set its **Display Flag** then set: **Bend** to **75, Capture Origin** to **-40, 0, 0, Capture Direction** to **-1, 0, 0,** and **Capture Length** to **20. RMB-click** on the *grid* node's output and type **Subdivide.** Set its **Display Flag** then set **Depth** to **2.**

CACHING OUT SIMULATIONS

To support the look dev stage of this project, you cached out the **fluid**, the **wine glass** and the **bullet** to geometry (USD) sequences. By doing this, you can focus on rendering without worrying about simulations being recalculated.

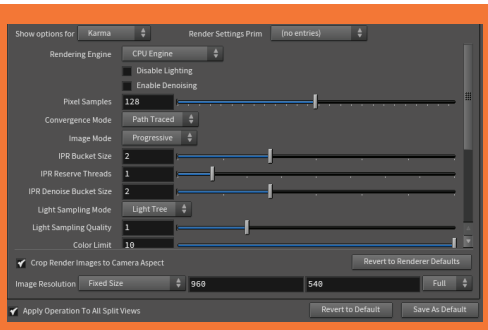
This is a typical workflow when working on VFX shots that can consume **LOTS OF HARD DRIVE SPACE**. You should be aware of this before sending off huge simulations with lots of particles - make sure you have somewhere to store the various intermediate stages.

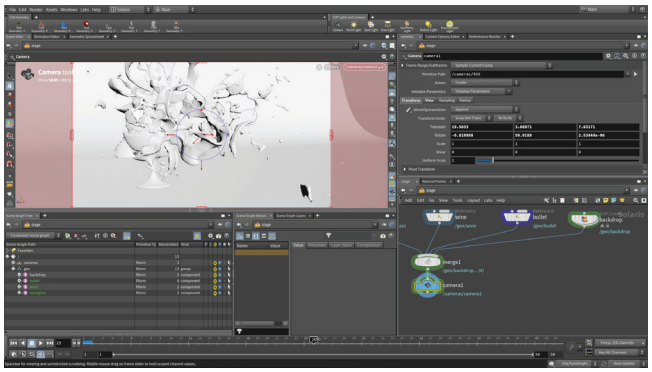


VIEWPORT RENDERING

You are now going to render the sequence using Houdini's renderer Karma. At first this will render using settings you can find in the Display options. Press **d** in the Scene View to bring it up. You can turn on the denoiser here, set Pixel Samples and the Image Resolution.

Later when you set up a **Karma LOP**, there will be render settings on that node which will be used to create the final output.





04 Use your view tools to look at the *wineglass* from the front. From the **LOP Lights and Camera** shelf, **Ctrl-click** on the **Camera** tool. This adds a camera node into the network and you are now looking through the camera in the viewport.

Press the **Lock Camera/Light to View** button so that view changes can be used to reposition the camera. Now **Tumble**, **Pan** and **Dolly** in the viewport to reposition the camera so the wineglass is on the left and the splash moves to the right. Scrub the timeline to make sure the camera works for the whole sequence.

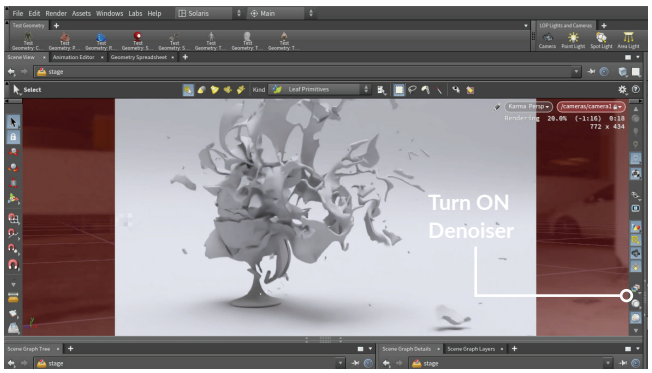
05 From the LOP **Lights and Camera** shelf, **Ctrl-click** on the **Environment Light** tool. This adds a *domelight* node to the end of the chain.

Select the *domelight* node and from the **Base Properties** tab, click on the **File Chooser** button next to **Texture**. Click on the *$HFS/houdini/pic/hdri* listing in the sidebar then select the *HDRIHaven_skylit_garage_2k.rat* file. Click **Accept**.

On the **Display Options** bar. Click on the **High Quality Lighting with Shadows** button.

06 From the *Persp* menu, choose **Karma** to render with Karma in the viewport. You can move to different frames in the timeline and the viewport will update quickly.

Karma is designed to work with USD which is why everything in the LOP context is converted to the USD scene graph. You can only use the Karma renderer from this part of Houdini.

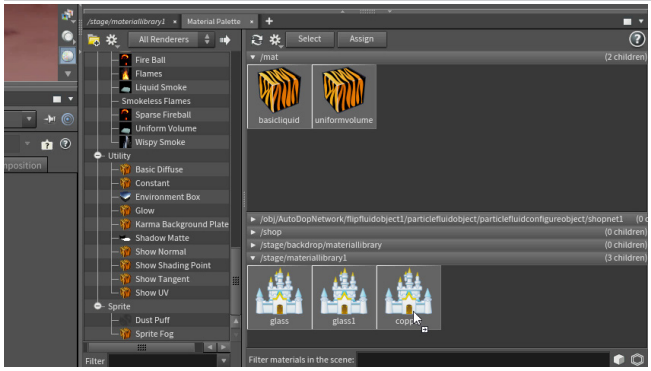


07 To get a cleaner image when you render, you can turn on **Denoiser** if you have an Nvidia graphics card and you have installed the latest drivers. You can turn it on in the **Display Options** bar.

PART EIGHT

Assign Materials and Render a Sequence

Now you can add materials to the wineglass, wine and bullet. These materials will become part of the USD Scene Graph and will get assigned to the geometry using a LOP node. You can then use a Karma LOP to prepare your render settings including the use of the Nvidia Optix Denoiser. After you render, you can load the sequence into Mplay to review the results.

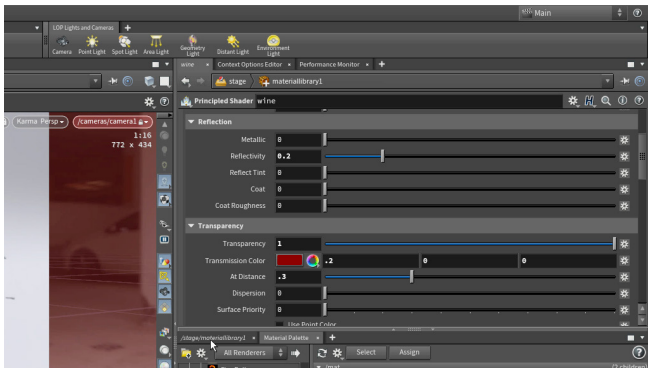


01

In the Network view, press **tab > Material Library. Wire** it into the end of the chain then set its **Display Flag.**

Go to the **Material Palette** pane. Click on the arrow next to */stage/materiallibrary* to open up this area. Scroll through the material gallery on the left of the palette and drag two **Glass** materials into the *materiallibrary* working area for the wineglass and wine.

Now find a **copper** material and drag it into the *materiallibrary* working area. You will use this for the bullet.

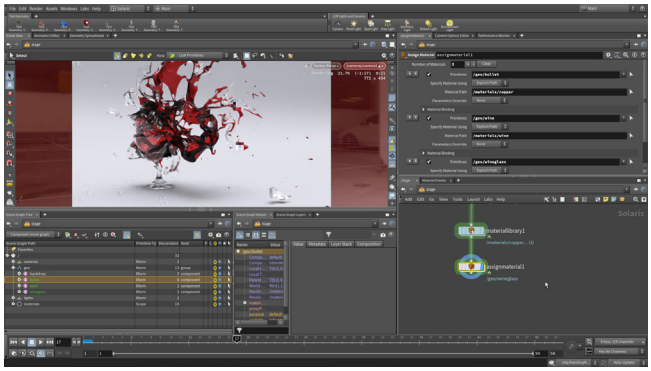


02

Select the second *glass* material and rename it to *wine*.

Set **Inside IOR** to **1.3443** which is an IOR for wine. Next, set **Reflectivity** to **0.2** to reduce how much the surroundings are reflected.

Set **Transmission Color** to **0.2, 0, 0** which will create a reddish wine look. Next set **At Distance** to **0.3.**



03

Go back to the **Stage** level. After the *Material Library* node, add an **Assign Materials** node.

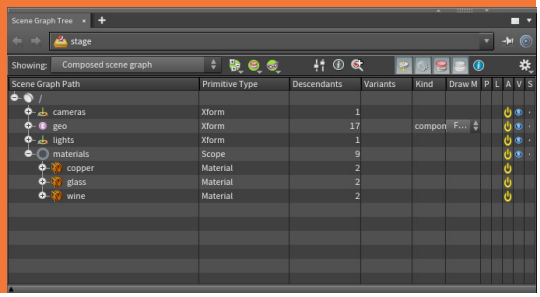
From the Scene Graph, drag the *wineglass* to the **Primitives** field then click on the arrow next to **Material Path** and choose the *glass* material for this primitive.

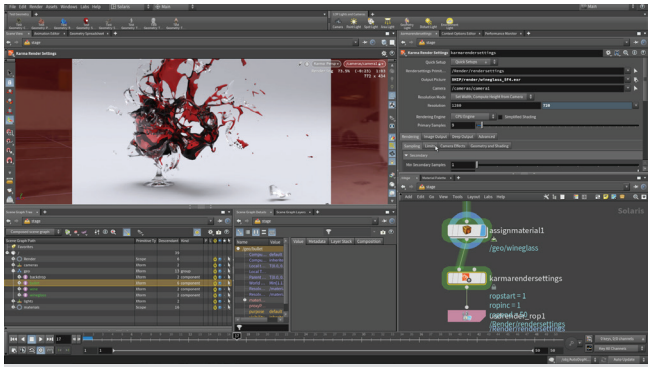
Now click the **Plus Sign** next to the check box twice to add two new entries. Use the same method to assign the *wine* material to the *wine* primitive and the *copper* material to the *bullet* primitive.

SCENE GRAPH

Just like the geometry and lights, the materials you add using LOP nodes are added to the **Scene Graph**. When you used the Material Library LOP, the default setting for the Material Path Prefix was **/materials/** and that is where they are placed in the graph. You could choose to organize them differently but this is the default.

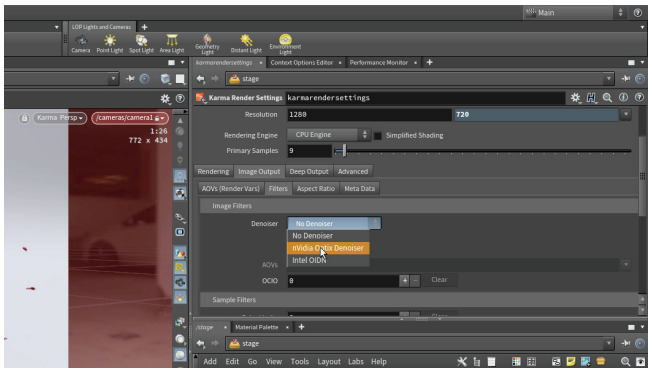
This material path is the one you used in the **Assign Materials LOP** to place the materials onto the geometry.





04 In the **Network View**, press **tab > Karma** to add a **Karma Render Settings** and **USD Render ROP** node. Wire them into the end of the chain. Select the *karmarendersettings* node and on the **Image Output > Filters** tab set **Denoiser** to **nvidia Optix Denoiser** to turn the denoiser back on.

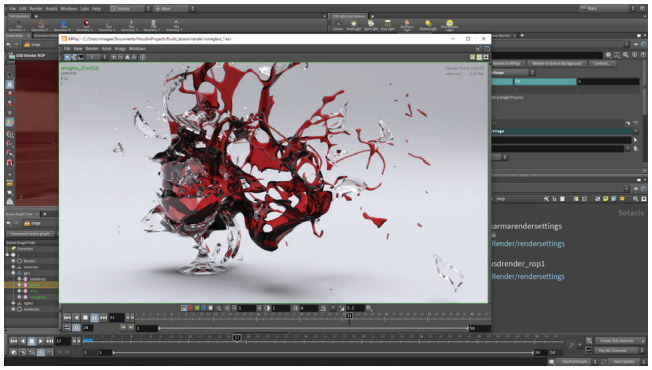
Select the *usdrender_rop* node. Set **Valid Frame Range** to **Render Frame Range** and set the **Output Picture** to *$HIP/render/wineglass_$F4.exr*. The *$F* in the name is needed to add frame numbers to the renderings and the *4* is the padding of the frame number.



05 The denoiser from the viewport will not affect the output from this node therefore you much choose it explicitly. Select the *karmarendersettings* node and on the **Image Output > Filters** tab set **Denoiser** to **nvidia Optix Denoiser** to turn the denoiser back on.

The **nVidia Optix Denoiser** will match the one used in the viewport. There is also an **Intel OIDN** denoiser which is only available when rendering to disk.

Save your work. Select the *usdrender_rop* node and click on **Render to Disk**.



06 When you finish, choose **Render > Mplay > Load Disk Files** and open up the rendered images to review the final sequence.

Later you can branch off another **Karma** node to up the resolution and render settings for your final rendering. You can go back to **Convergence Mode** set to **Variance,** up the **sample** count and turn off the **denoiser**. It is always good to complete test renderings at a lower resolution first to make sure that everything is working the way you expect it to.

CONCLUSION

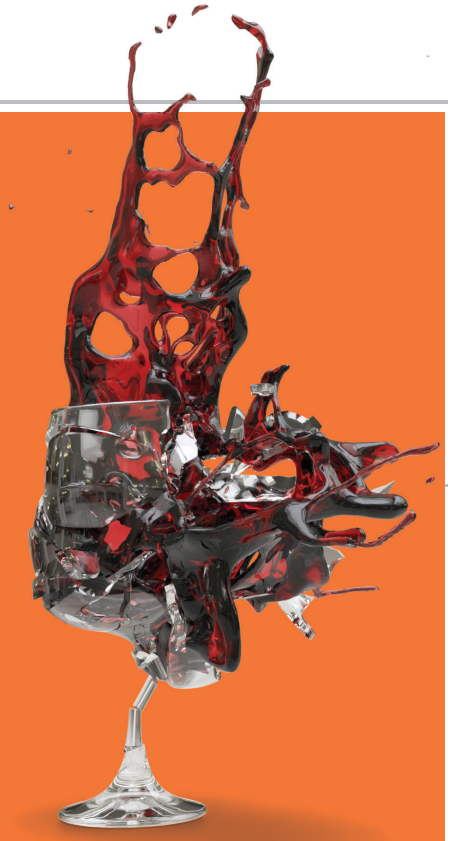
You have now created a complete VFX shot using the Bullet RBD and FLIP Fluid solvers to smash a wine glass. You used the retime node to slow down then reverse time so that the wine glass finished in the same position as it started then you cached out the results to USD files.

These were then used to set up and light your shot using the Solaris/LOPS context. Materials were created and assigned to the primitives get the right look for your shot.

This project shows how you can use Houdini's dynamic nodes and networks to integrate different kinds of effects while using geometry nodes to set up and output the simulations.

Now that you have an understanding of the nodes and networks used to create VFX shots. This will assist you as you dig deeper into Houdini to achieve your own effects.

Enjoy!



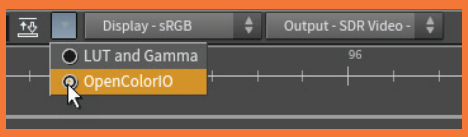
HOUDINI FOUNDATIONS

DESTRUCTION FX

One of the things that makes visual effects fun is that you get to blow things up without causing any real damage. In this lesson, you will light a fuse using particle sparks then explode a cartoon bomb using rigid body dynamics for the shell of the bomb and Pyro FX for the fire and smoke. This lesson will teach you how to set up dynamic simulations using a variety of shelf tools and network nodes.

To give you a complete understanding of the shot being developed, you will build all the elements from scratch, then simulate the effects. This will help you understand how the simulation nodes work within the wider context of a Houdini scene. In the end, you will render out the shot using the Karma renderer.

ACES | OPENCOLORIO SETUP



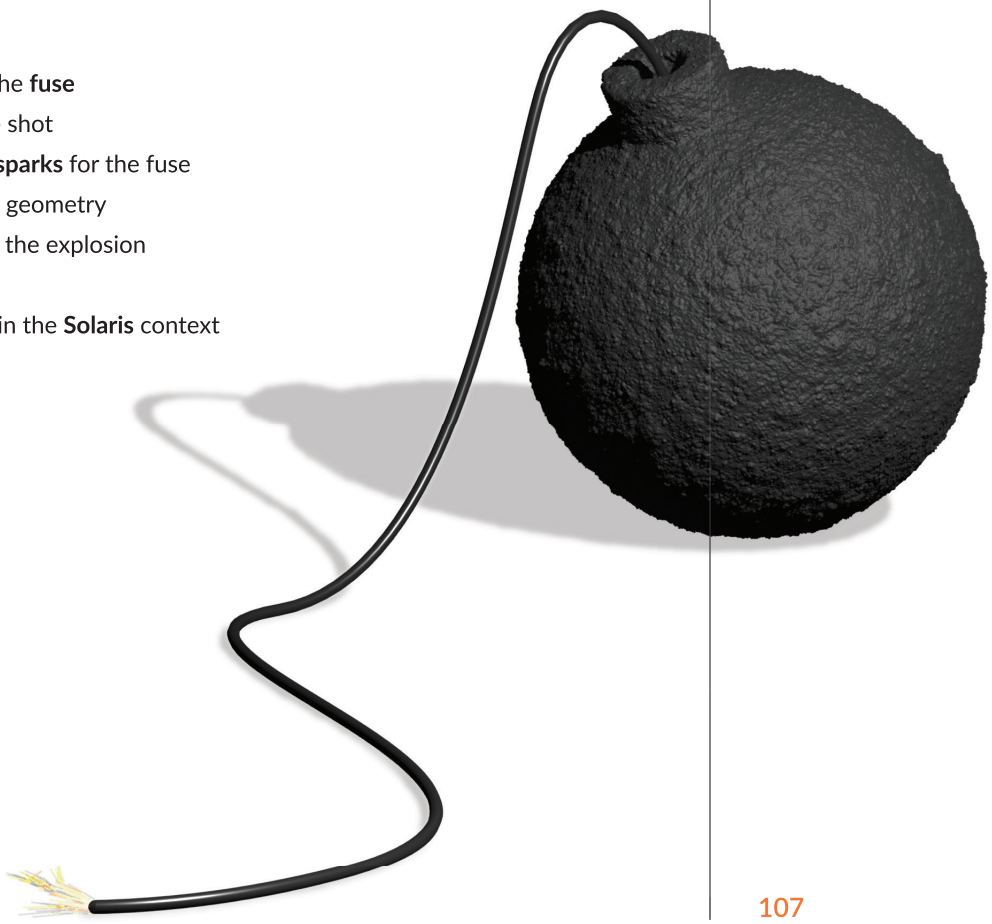
For more accurate color display when working with Pyro FX, you should use the **Academy Color Encoding System (ACES)**. To use it, bring up the **Correction Toolbar** from the **viewport (persp)** menu in the scene view. From the arrow button on the right, choose **OpenColorIO**. This will give you a Display of **sRGB** and an output of **SDR Video - ACES 1.0**. This setting only works for your current session and will need to be turned back on each time you open Houdini.

LESSON GOAL

Model then blow up a bomb using particle sparks, rigid body dynamics and Pyro FX.

WHAT YOU WILL LEARN

- How to model the **bomb** and animate the **fuse**
- How to **animate a camera** to set up the shot
- How to set up a **particle soot trail** and **sparks** for the fuse
- How to **shatter** then explode the bomb geometry
- How to set up a **Pyro FX** simulation for the explosion
- How to set up **materials** and **textures**
- How to **render** the effects with **Karma** in the **Solaris** context



LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

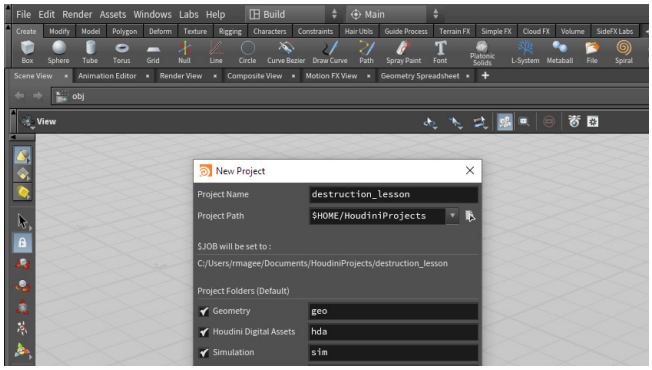
Houdini Core	✘
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 1.0 | July 2022

PART ONE

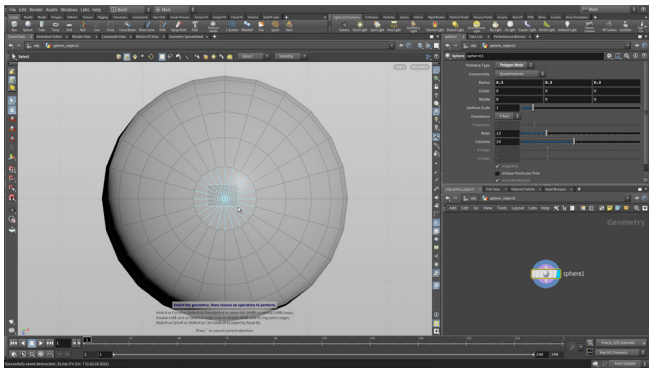
Model the Bomb

To create the bomb geometry, start with a primitive sphere and modify it to define the opening at the top. This will involve a few poly extrudes and bevels to define the geometry you will need for the final shape. Later in the lesson, you will fracture the bomb.



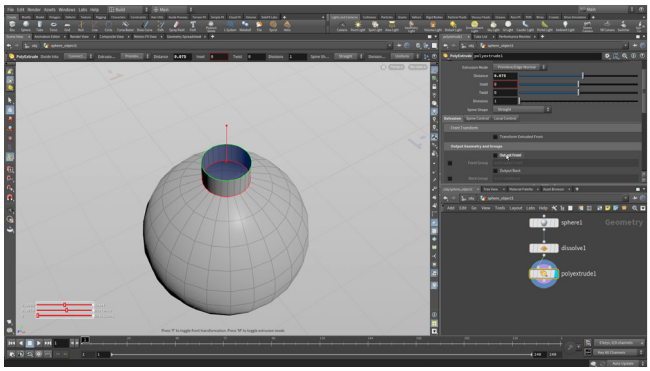
01 Select **File > New Project.** Change the **Project Name** to *destruction_lesson* and press **Accept**. This creates a project directory with sub directories for all the files associated with this shot.

Select **File > Save As...** You should be looking into the new *destruction_lesson* directory. Set the file name to *destruction_01.hip* and click **Accept** to save.



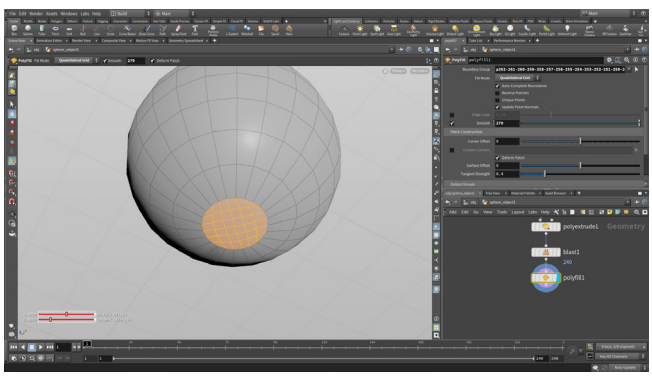
02 In the viewport, **press c** to bring up a radial menu. From this menu, choose **Create > Geometry > Sphere**. In the viewport, press **Enter** to place it at the origin. In the Operation Control bar at the top, set **Radius** to **0.3, 0.3, 0.3**.

Press s to get the select tool then **3** to invoke edge selection. Press **Spacebar 2** to go to a Top view. Box select the edges at the top and bottom of the circle and press **delete**. This dissolves the edges and leaves two circular polygons in their place Press **Spacebar 1** to go back to a perspective view.



03 **Press s** to get the select tool then **4** to switch to primitive (face) selection. Select the circular polygon at the top of the sphere.

Press c to bring up the radial menu and select **Model > Polygons > Polyextrude.** Move the handle up by a **Distance** of about **0.075.** Set **Output Front** to **Off.**



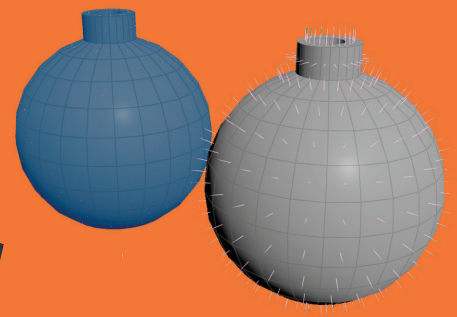
04 **Select the** circular polygon at the bottom of the sphere and **press Delete.** This will add a **blast** node to the network. **Press s** to go to the **Select** tool and **3** to change to edge selection. **Double click** on the edge of the hole you just created to select all the edges.

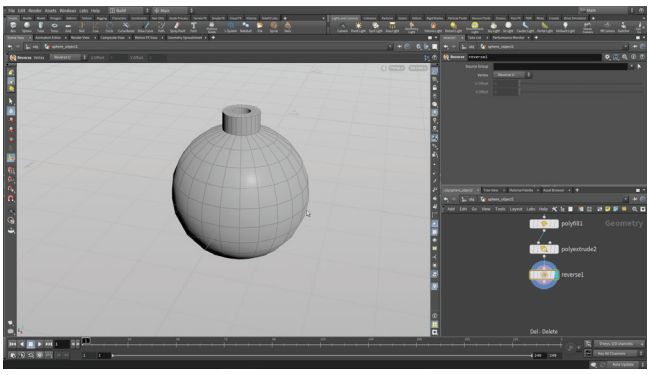
Press **tab > Polyfill**. In the Parameter pane, set **Fill Mode** to **Quadrilateral Grid** and **Smooth** to **270**. This creates a cleaner topology for the bottom of the sphere that doesn't go to a single point.

SURFACE NORMALS

Every primitive has a normal direction where one side is the inside and one is the outside. When you polyextrude the bomb geometry it will be inside out at first. This is indicated with the blue color on the faces. You can then use a Reverse node to redirect the normals.

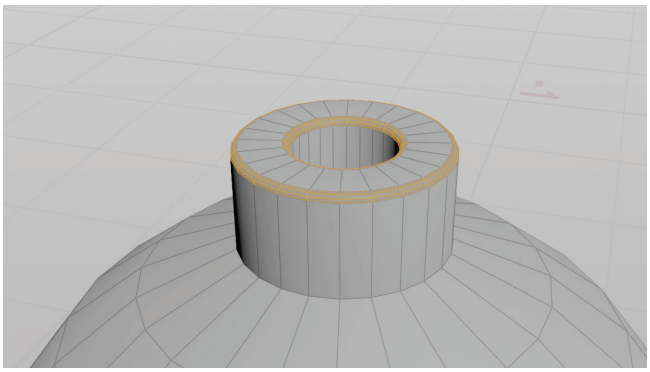
You can see the normals on the surface using the **Display Primitive Normals** button found in the **Display Options** bar on the right side of the **Scene view** pane.





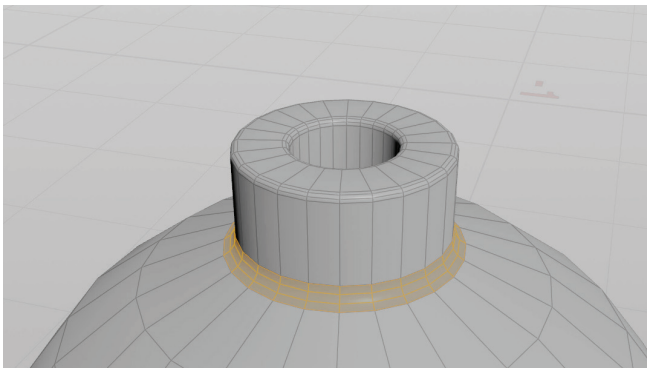
05 **Press n** to select all Faces and again get the **Polyextrude** tool. Extrude to a **Distance** value of around **-0.04.** In the Parameter pane in the **Extrusion** tab, turn on **Output Back**.

Press n to select all Faces and again press **tab** and start typing **Reverse**. This node reverses all the polygon normals. Now they are pointing in the right direction.



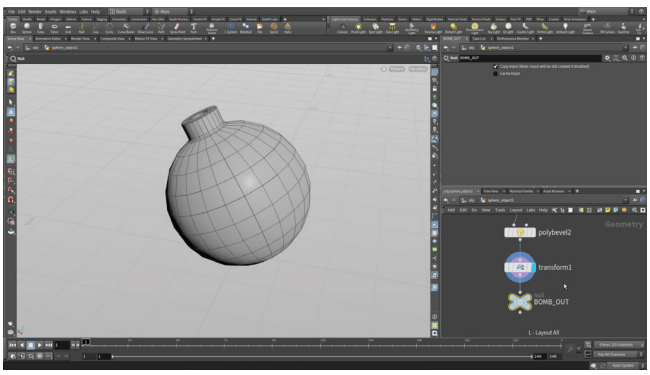
06 **Press s** to go to the **Select** tool and **3** to change to edge selection. **Double click** on the edge at the top of the bomb and then press **Shift** and **double click** to select the inner circle at the top.

Press c to use the radial menu to go to **Model > Polygons > Polybevel**. Set **Distance** to **0.005**. Set the **Shape** to **Round** and **Divisions** to **3**.



07 **Press s** to go to the **Select** tool. **Double click** on the edge where the circular part of the bomb meets the extruded section at the top.

Press q to repeat the last tool which was **Polybevel** and set **Distance** to **0.01**, Shape to **Round** and **Divisions** to 3.



08 In the Network view, press **tab > Transform** and add it to the end of the network. Set **Translate Y** to **0.3** and **Rotate X** to around **27** degrees.

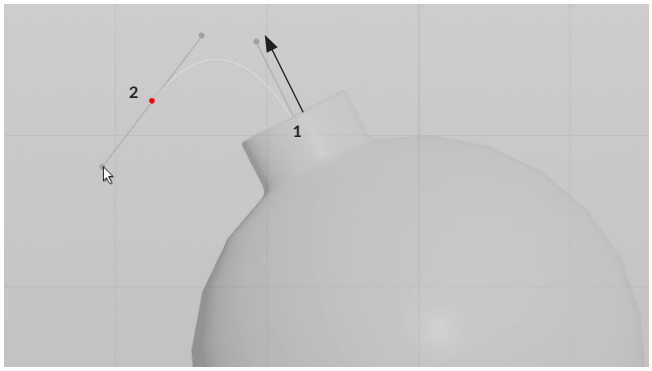
Add a **Null node**. Wire the end of the *polybevel* to the *null* then set the **display flag** on the *null* node to display it. **Double click** on its name and change it to *BOMB_OUT*.

Go to the **Object** level and rename the object to *bomb_geo* since it holds the bomb's geometry.

PART TWO

Model the Fuse

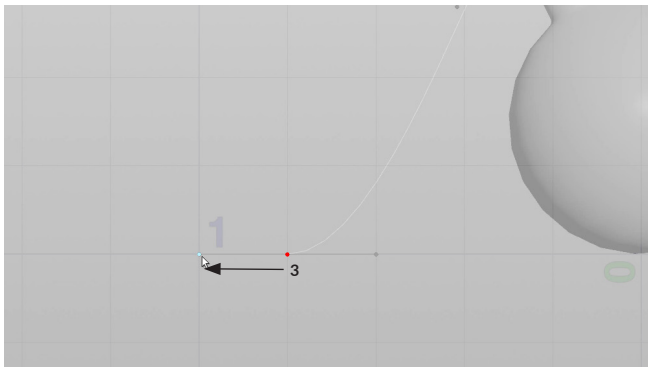
To create the Fuse, start with a Bezier curve that emerges from the top of the bomb. You can then snake the curve on the ground to create a longer fuse. Reverse the curve direction to get ready for animating the fuse then add a Polywire node to give the fuse thickness.



01

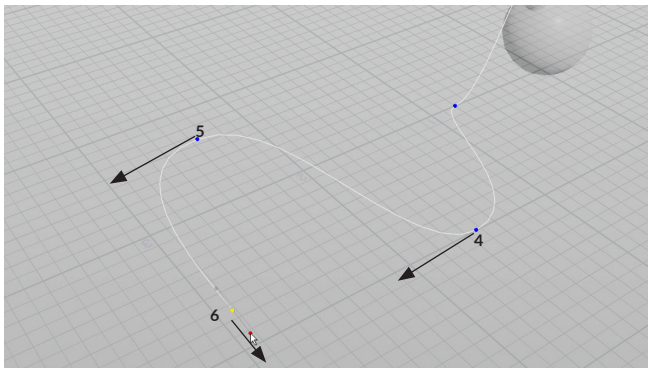
Press **spacebar-b** to see the bomb from all angles. Move over the **Right** view and press **spacebar-b** to expand it.

Press c to bring up the radial menu and choose **Create > Geometry > Curve**. **Click drag** up to create the first point and tangent handle for the curve. Next add a point and **drag down** to quickly draw the curve pointing down.



02

Press x and choose **Grid** to turn on grid snapping. Click and drag on the ground to create a third point with its tangent handle aligned along the ground.



03

Press **spacebar-b** to go back to 4 views then mouse over the *perspective* view and press **spacebar-b** again expand the view.

Turn off **Grid Snapping** then turn on the **Construction Plane** using the second button at the top of the **Display Options** bar. This will make sure that any edits you make stay on the ground.

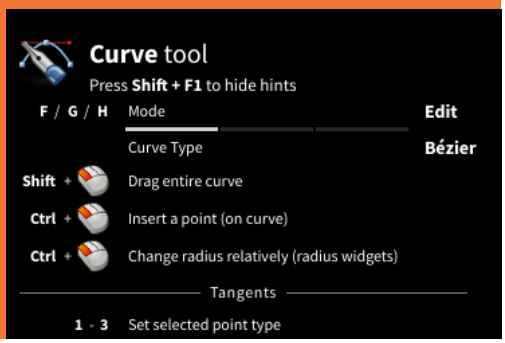
Draw three new points with their tangents dragged out to define the curve's shape.

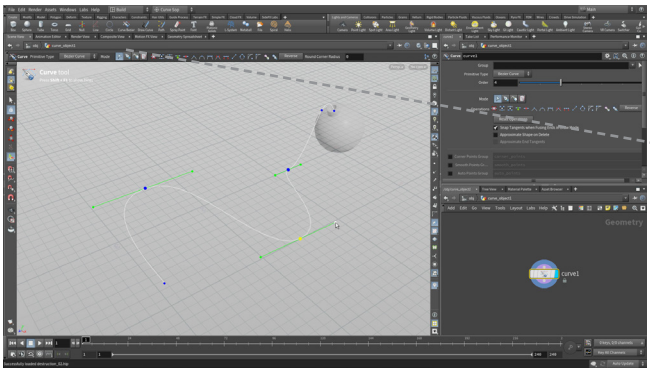
MMB-click to complete the curve.

TOOL HINTS

The curve tool comes with tool hints that display in the Scene view as you work. These provide different shortkey options for this tool and help you get familiar with how it works.

You can collapse it using **Shift + F1** and only the tool name will display. There are a growing number of tools that use tool hints and you will see more of them in future versions of Houdini.

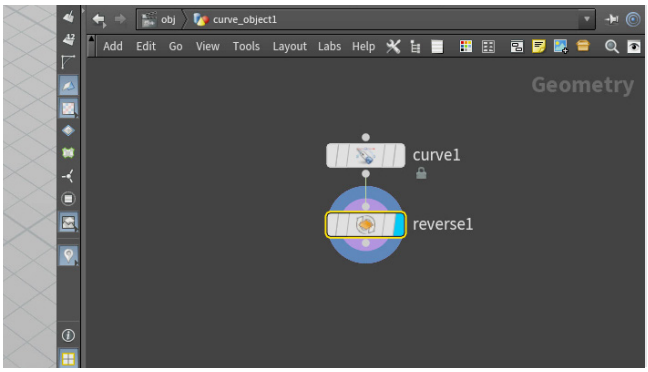




04

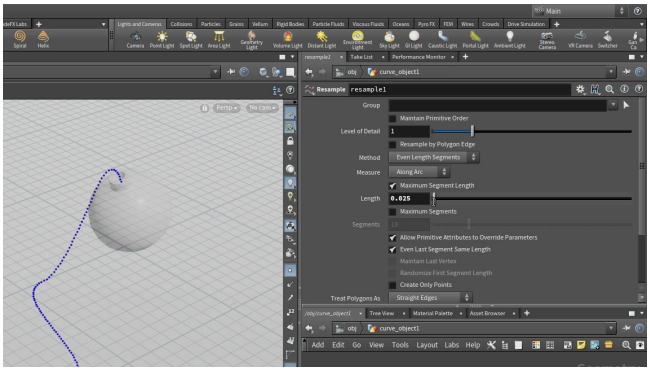
In the **Operation Control** bar, change the curve **Mode** to **Select/Edit**. Now you can click on the edit points on the curve and make changes to refine the shape of the curve.

Select and edit the points on the ground to get the desired look for the curve. Tumble around to make sure that your curve stays above the ground plane.



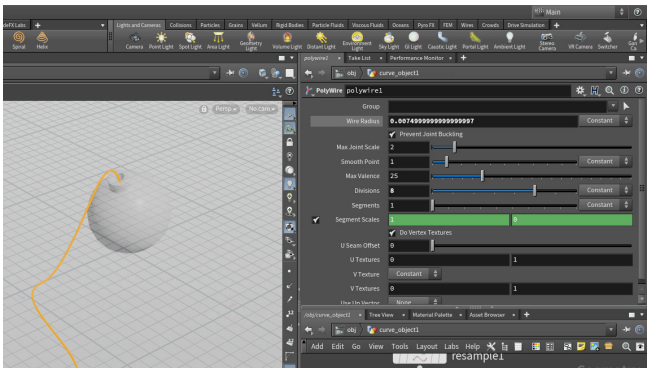
05

In the Scene View, press **tab > Reverse**. **Press n** to select the whole curve and press **Enter**. Since the curve was drawn from the bomb out, it will not animate in the direction you need. This will put the start of the curve where the fuse begins.



06

Turn on the **Display Points** option in the **Display Options** bar. Add a **Resample** node. On the *resample* node, set the **Maximum Segment Length** to **0.025** to add more detail. The *resample* node evens out the points.

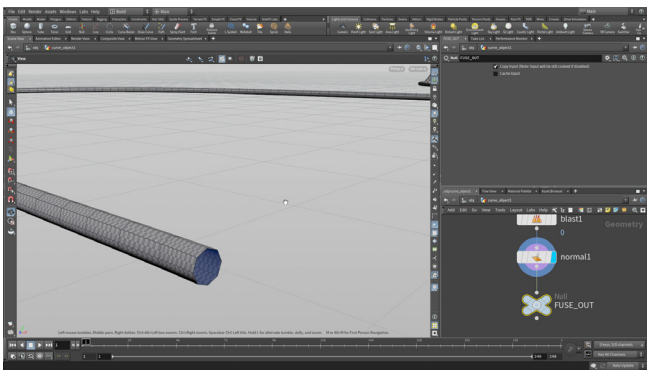


07

Add a **Polywire** node to add thickness to the wire. Set the **Wire Radius** to **0.0075** and the **Divisions** to **8**.

Between the *reverse* and *polywire* nodes, add a **Transform** node. Go to the polywire node and **RMB-click** on the **Wire Radius** parameter and choose **Copy Parameter**. Now go back to the *transform* node and **RMB-click** on **Translate Y** and choose **Paste Relative References.**

This will lift the whole fuse up so that it is not halfway under the ground grid.



08

Add a **Blast** node Set **Group** to **0**. This deletes the end of the fuse geometry. Next add a **Normal** node to the end of the chain.

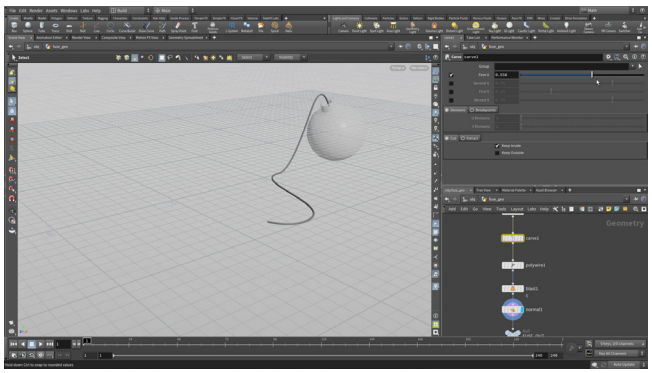
After the *normal* node, add a **Null** node and name it *FUSE_OUT*. This gives you a node representing the whole fuse geometry.

Go back up to the object level and rename this object to *fuse_geo*.

PART THREE

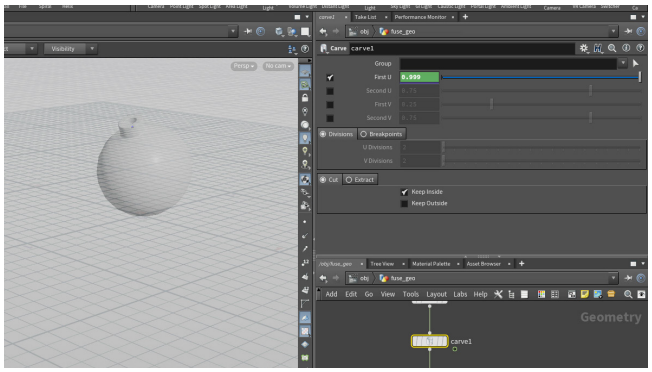
Animate the Fuse

Animate the fuse using a Carve node that lets you control the length of the curve over time. Add a round cap to the fuse that will be used to emit soot and sparks. You need to set up tangents on the curve to ensure that the cap follows along properly. Next add some NULL objects to make it easier to export the cap for use in emitting particles.



01 Dive back into the *fuse_geo* object. Between the *transform* node and the *Polywire* node add a **Carve** node. Drag on the **First U** slide to see how this affects the curve.

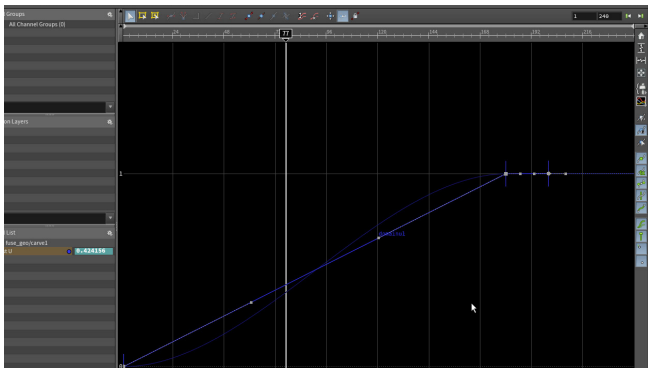
Set **First U** to **0**. **Alt click** on **First U** to set a keyframe at **frame 1**. The parameter box will change color to indicate that it has been keyframed and that there is a key at the current frame.



02 Go to **frame 180**. Set **First U** to **0.999**. This will set a keyframe. You can see that the fuse will. Go to **frame 200**. Set **First U** to **1.0**. This will set another keyframe.

In the bottom left of the Playbar, toggle **realtime playbark** on so that it doesn't play back too fast and press **Play**.

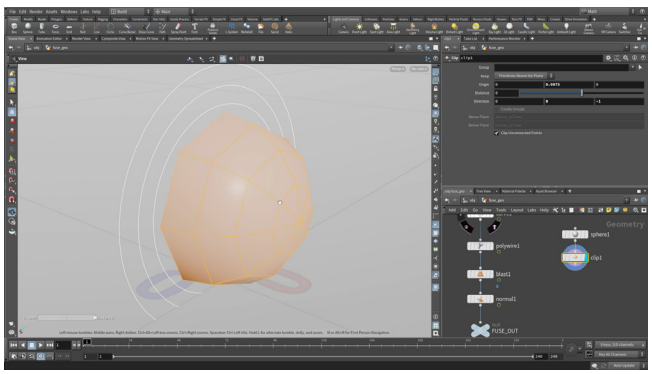
The fuse now animates into the bomb geometry where you will set up the explosion.



03 Click on the **Animation Editor** Pane tab. Select the animation curve and click on the **Straight** button at the top of the panel. This will straighten the curve and the fuse will animate evenly from start to finish instead of speeding up then slowing down at the end.

Go back to the Scene View pane tab and **playback** the animation to see how this change affects the motion.

Save your work.



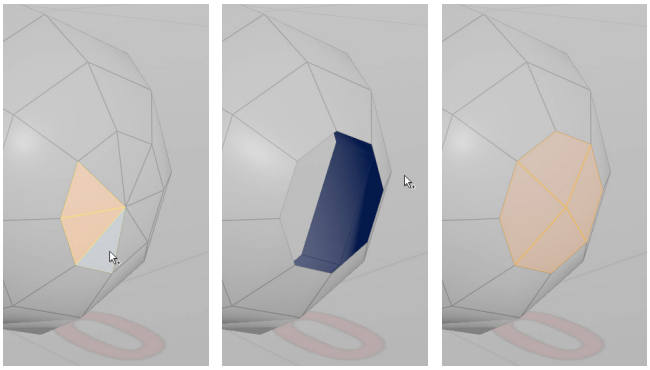
04 In the Network View, add a **Sphere** node and set its **Display Flag**. Set the following:

- **Radius** to **1, 1, 1**
- **Center Y** to **0.0075**
- **Uniform Scale** to **0.0075**

Press **Spacebar-F** to focus on it.

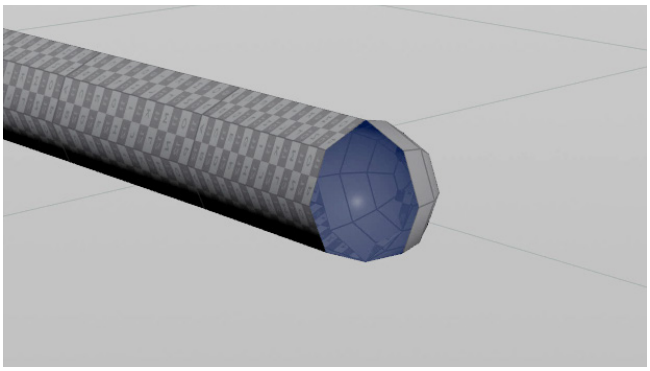
- **Orientation to Z Axis**
- **Rows** to **9** and **Columns** to **8**

In the Scene view, **press n** to select all then **tab>Clip** node and set its **Direction** to **0, 0, -1**.



05 Tumble around and **press s** to get the select tool and **4** to get **face/primitive** selection. Select one of the triangles at the tip of the sphere then **press and hold** the **a key** and then **middle click** two triangles over to select all the triangular faces. Press the **Delete** key to remove them. This adds a *blast* node to the network.

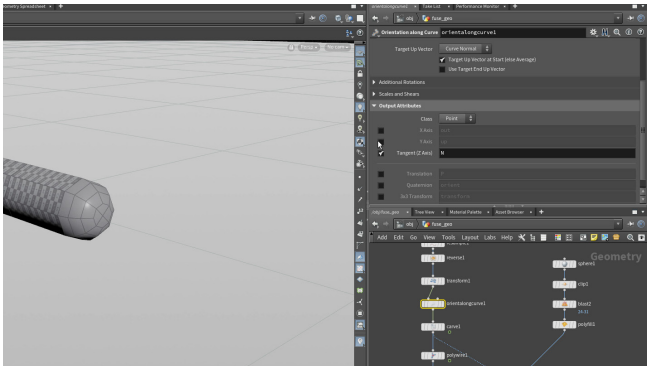
Press 3 to go bring up edge selection then double click on the edge of the area you just blasted. Press **tab > polyfill** which places the node after the *blast*. Set **Fill mode** to **Quadrilateral Grid** then turn on its **Display flag**. Set **Smooth** to **100** and **Tangent Strength** to **0**. This will create quad topology at the tip of the sphere.



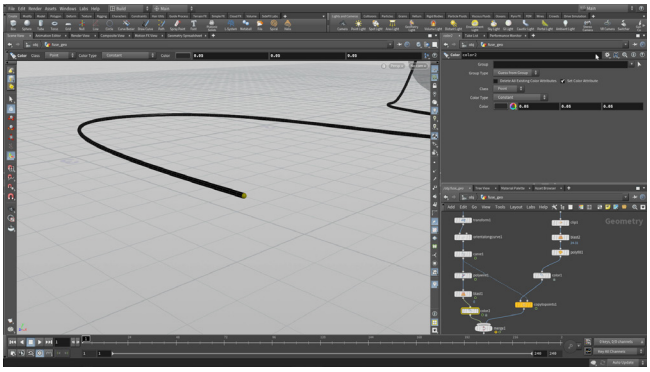
06 Add a **Copy to Points** node into the network. Feed the *polyfill* node from the sphere into the first input and the *carve* node into the second input. Set **Target Points** to **0**.

Add a Merge node. Feed the *blast* node and the *copytopoints* node into it then wire it into the *normal* node.

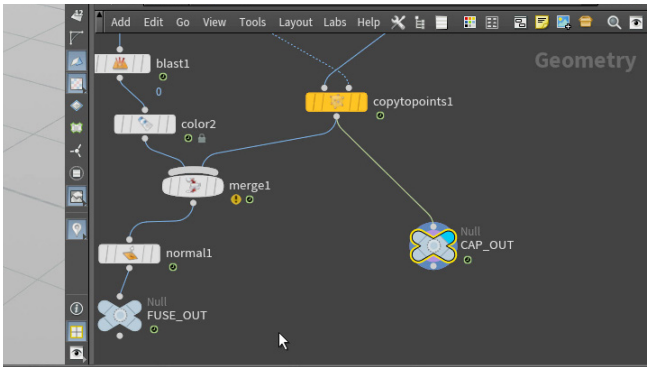
The cap it positioned properly at the end of the curve but it isn't oriented correctly. You need to add normals to the curve to allow for proper alignment.



07 Add a **Orientation Along Curve** node between the *reverse* and the *carve* nodes. Under **Output Attributes** turn off the **Y Axis** option. Leave **Tangent (Z axis)** set to N. This adds normals to the curve that will align the end cap as it moves along the fuse.



08 Add a **Color** node after the *polyfill* node and set the Color to **yellow**. Add another **Color** node after the fuse's *blast* node and set its **Color** to **Dark Gray**. These colors will be helpful for visualizing the fuse as you work and can also be used to affect the materials being assigned later.



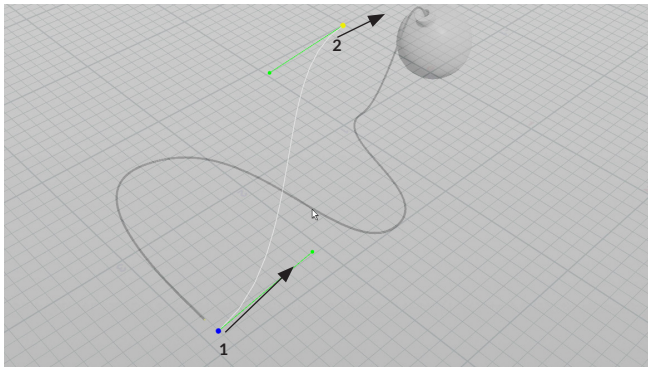
09 Branch off a **Null** node after the *copytopoints* node and place it to the side. Name the *null* node *CAP_OUT*. You will use this later to extract the cap into another network that you will reference to emit particles. Set its **Display Flag** to see only the half sphere. You can scrub the **Playbar** to see it move with the carve.

When you are finished, set the **Display Flag** back to the FUSE_OUT null.

PART FOUR

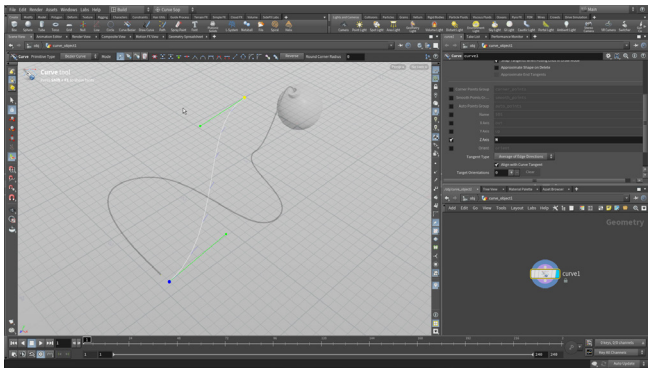
Create an Animated Camera

As you develop this shot further, it would be helpful to have a camera set up to frame the final shot. This camera rig will be built by constraining a null object to a curve then using an aim constraint to point the camera to the null object. This will give you a camera that follows the end of the fuse to make it easier to evaluate the particles as they are being emitted.

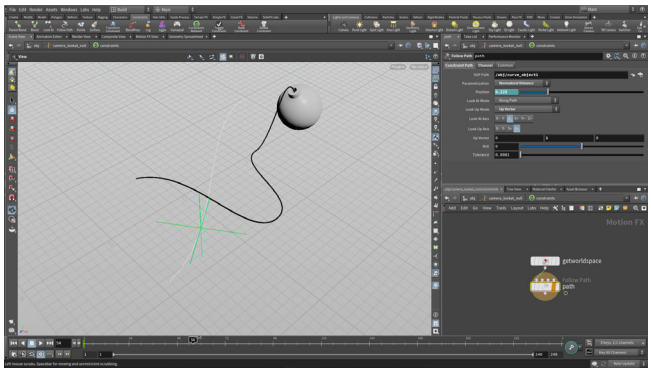


01 Dolly out to see the whole scene from above. Make sure the **construction plane** is **on**. **Press c** to bring up the radial menu and choose **Create > Geometry > Curve**. **Click-drag** a point near the start of the fuse and drag forward to extend the tangent.

Next, **click-drag** a second point behind the bomb and drag to pull out the tangent to create an s shaped curve. **MMB-click** to finish and use **Select/Edit** mode if you want to tweak the shape.

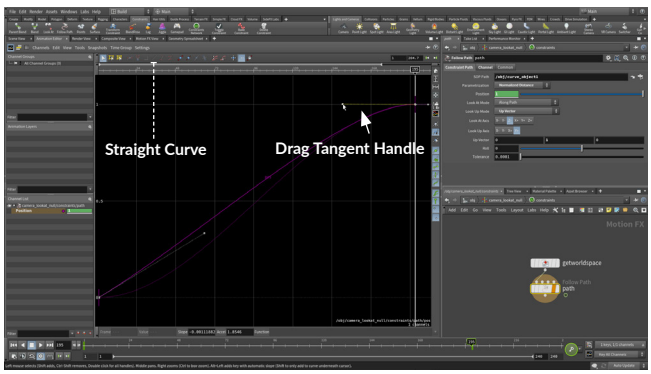


02 On the Curve node, turn on the Z Axis option and set it to N. This will create normals that will assist with animation along the curve.



03 Add a **Null** object at the origin. Rename this node *camera_lookat_null*. From the **Constraints** shelf, click on the **Follow Path** tool. This accepts the null as the starting object. Select the *curve* as the path object then press **Enter**. Press **Enter** twice more since you don't need a **look-at** object or a **look-up** object.

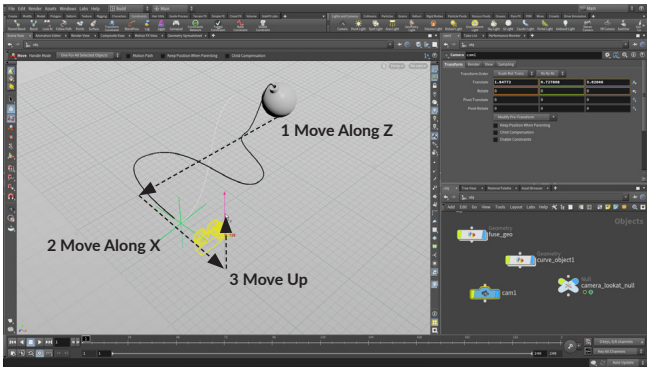
Now if you scrub in the timeline, you can see that the null object is moving along the Path from the first frame to the end frame at an even pace.



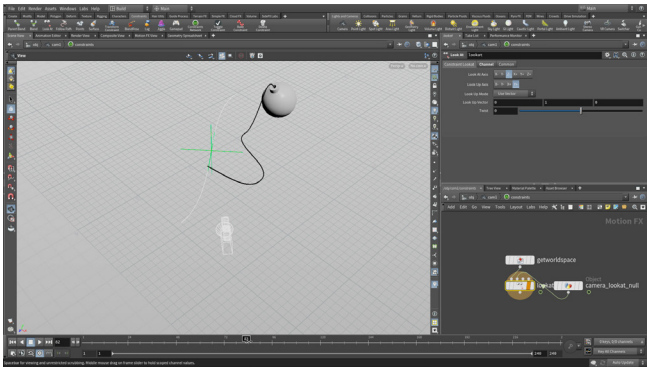
04 A constraint network was added to the null object and a path node is created. Go to **frame 1**. With the Path node selected, RMB click on the **Position** parameter and choose **Delete Channels** to remove the expression that is animating the null along the path. Set **Position** to **0**. **Alt click** on **Position** to set a keyframe.

Go to frame 195 and set **Position** to **1**. **Alt-click** on **Position** to set a second keyframe.

Click on the **Animation Editor** tab and **press h** to see the whole curve. Select it and press the **Straight** button to make it linear. Grab the tangent on the second point and adjust to smooth out the end.

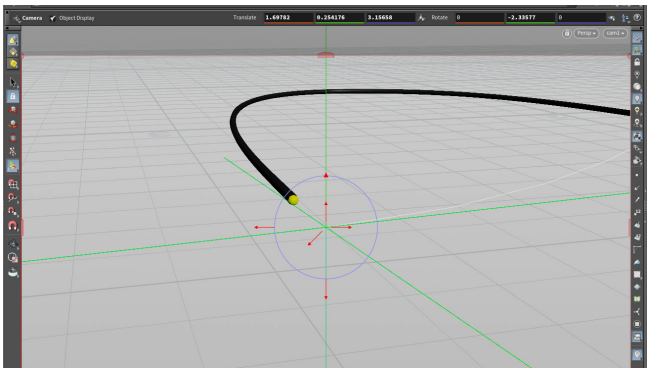


05 In the Network view, press **tab > Camera** and press **Enter** then click to place it at the origin. Now use the **Move** tool to move it in front of the fuse and a bit to the right. Next move it up along the Y axis to raise it from the ground by about **0.75** units.



06 From the **Constraint** shelf, click on the **Look At** tool. This will use the selected Camera as the look at object. Select the *null* object as the **look at** object and press **Enter**. Press **Enter** again to not assign anything as the **look up** object.

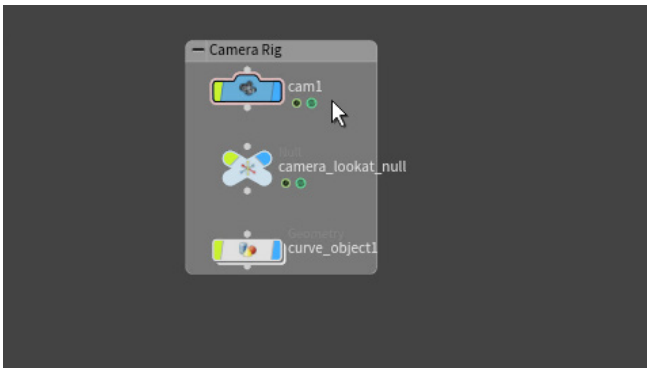
Now the camera is looking at the Null object. From the Camera menu, select *cam1* to look through this camera.



07 Go to **Frame 1**. Select the *camera* and make sure the **Handle** tool is active. This brings up a camera handle which you can use to reposition the camera to have a better view of the fuse's starting point.

Go to **Frame 195**. Use the same handles to reposition the camera to make sure the bomb in nicely positioned in the frame.

You may need to scrub back and forth to tweak the camera to work properly throughout the sequence. Make sure you can see the fuse throughout the whole sequence.



08 Go to the object level and select the *curve*, *camera_lookat_null* object and *cam1* node and align them then put them into a network box. Double click on the box's title bar and name the box *Camera Rig*. **Turn off** the display of all the parts so that you don't see them in the scene view as you work. You can collapse this box if you want or keep it open if you want to work with it further.

Save your scene.

CONSTRAINTS

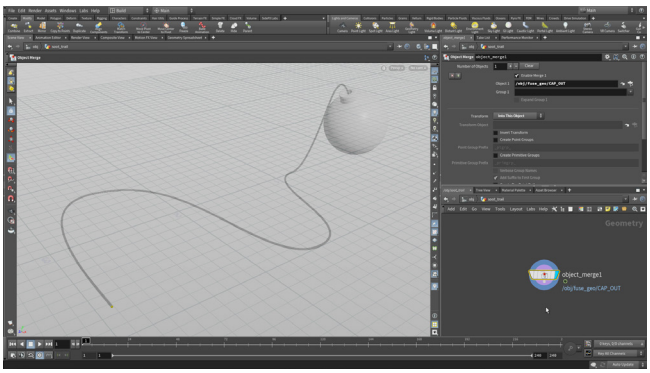
To get the null object to follow the path and then to get the camera to look at the path, you used animation constraints found on the **Constraints** shelf. The are accomplished using a special node type called **Channel Operators** or **CHOPS**. You can find these nodes inside the null and camera nodes. You can use these to control how the constraints work.

Another way to work with CHOP nodes is to use the **Motion FX** menu that you find when you **RMB-click** on any parameter.

PART FIVE

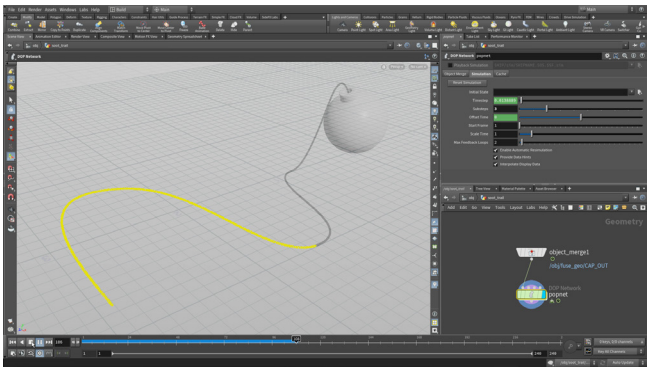
Create a Soot Trail

To create a soot trail, use the end cap to emit a trail of particles. Learn how to emit these points properly and how to add forces such as gravity to control the motion of the particles. Learn how to set up collisions where the particles either stick to the ground or slide off the bomb surface.



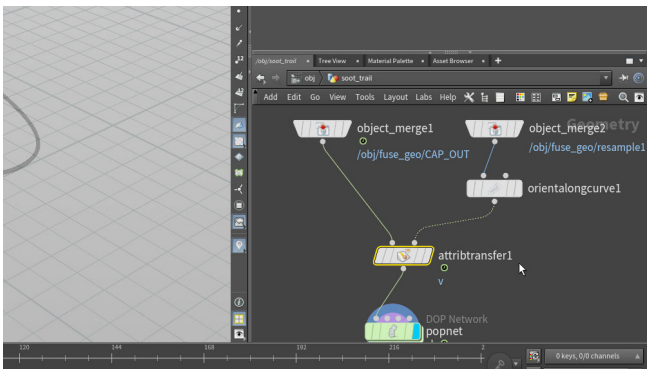
01 Dive back into the *fuse_geo* object node. With the display flag on *CAP_OUT*, go to the **Modify** shelf and use the **Extract** tool. **Press n** to select all the faces then press **Enter** to create a new object with the Cap being imported using an *Object Merge* node. Go up to the object level and name this object *soot_trail*.

Go back to the *fuse_geo* object and set the display flag to FUSE_OUT. Now you will see the combined fuse geometry when you render and the new object will be used to generate particles.



02 Dive into the *soot_trail* node and add a POP network node to the end of the chain. Dive in and on the *Source First Input* node, set **Const Birth Rate** to **1000**. Press **Play**. You can see particles being emitted but they aren't doing anything.

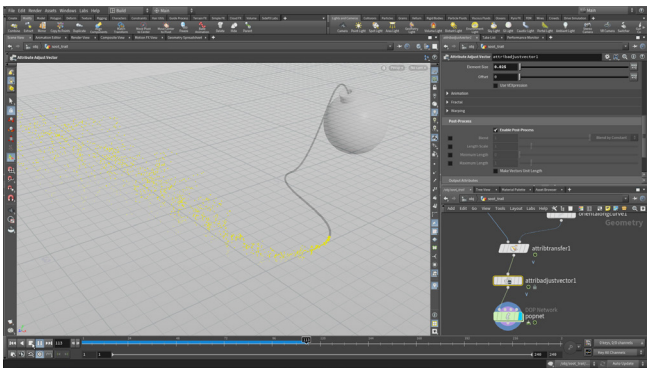
Go back up one level and on the **Simulation** tab, set **Substeps** to **3**. Press **Play**. You can see particles being emitted more evenly.



03 **Alt-drag** on the *object_merge* node to make a second copy. Set **Object 1** to the *fuse_geo>resample* node. You are going to use this curve to transfer velocity onto the cap.

Now add a **Orient Along Curve** node and under **Output Attributes** turn on **Tangent (Z Axis)** and set it to **v**.

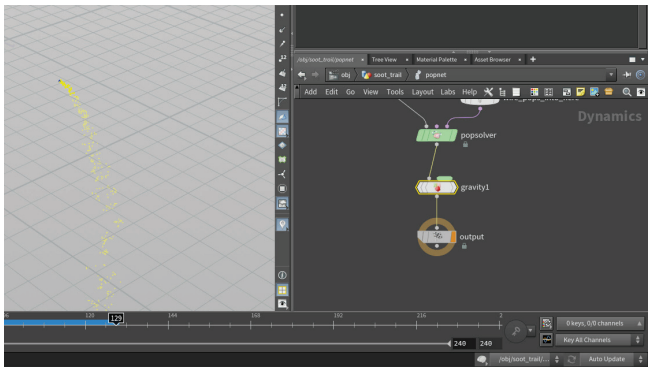
Add an **Attribute Transfer** node between the original *object_merge* and the *popnet* node. Turn off the **Primitives** checkbox and set **Points** to **v**.



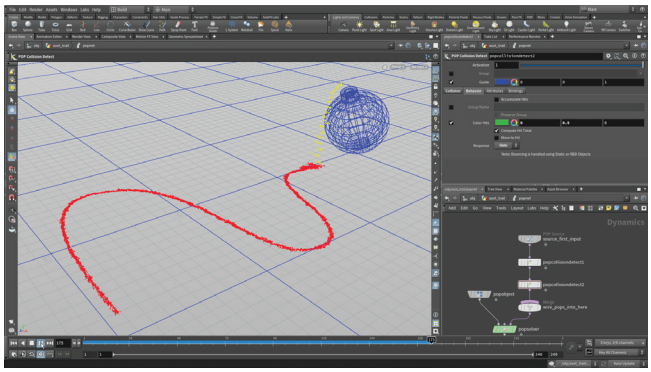
04 Place an **Attribute Adjust Vector** node in between the *attributetransfer* and the *popnet*. Under Adjustment value set the following:

- **Adjustment for** to **Direction Only**
- **Adjust with** to **Noise**
- **Range Values** to **Zero Centered**
- **Amplitude** to **0.5**

Under **Noise Pattern** set **Element Size** to **0.025** and under **Post Process**, turn on **Enable Post-Process**. Press **Play** to test.



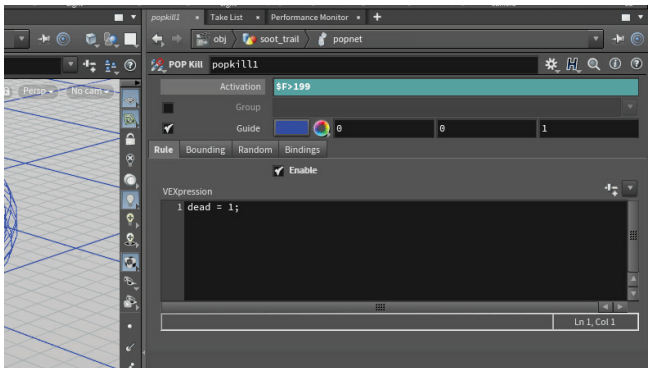
05 Dive back into the *popnet* and add a **Gravity Force** node under the *popsolver*. Now if you **play** the simulation the particles fall below the ground.



06 Go back up to the Object level and create a **grid**. Set **size** to **30, 30** and **Rows and Columns** to **31, 31.**

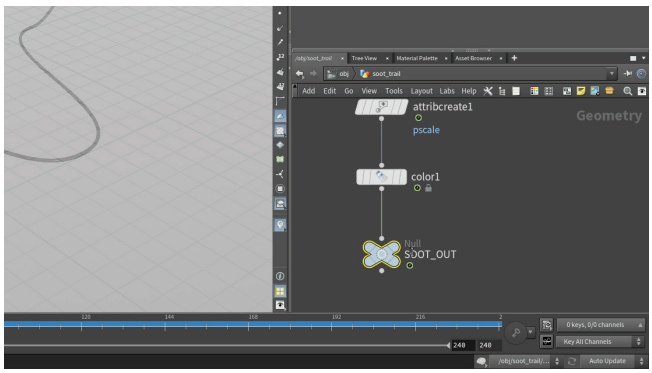
Rename this to *ground*. Go back into the *popnet* and add a **Pop Collision Detect** node after the *source_first_input* node. Set the **SOP** Path to the `obj/ground/grid1`. On the **Behavior** tab, set **Response** to **Stick**. Keep **Color Hits** set to **Red**.

Add another **Pop Collision Detect** node. Set the SOP to the *bomb_geo* geometry object. Set **Response** to **Slide**. Change **Color Hits** to **Green**. Press **Play**.



07 Right now the particles will be emitted throughout the whole sequence. You need them to stop them just as the bomb explodes. Add a **POP Kill** node after the *wire_pops_into_here* merge node. Go to frame 1. Set **Activation** to **$F>199**. This will kill the particles at frame 200.

Now under the Rule tab, turn on Enable. Press **Play** to test.



08 Jump back up to the geometry context and wire the *popnet* into an **attribute create** node. Set **Name** to **pscale** and **Value** to **0.001**.

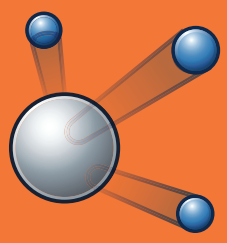
Add a **Color SOP** and set the **Color** to **dark grey**.

Add a **null** to the end of the chain and name it *SOOT_OUT*.

PARTICLE FX

The soot trail is created using particle dynamics. Particles are points that you can affect using forces such as wind or gravity. Starting with the end of the fuse, you give birth to points that are then simulated using a number of different techniques.

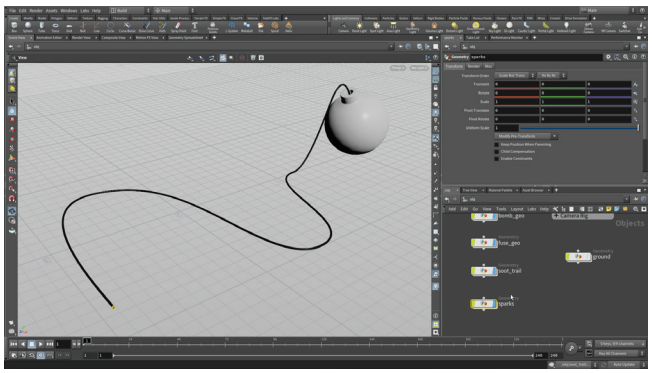
Particles are simulated using the Dynamics or DOPS section of Houdini then brought back into SOPS where you can work with them as geometry. In the next section, you will use particles to create sparks at the end of the fuse.



PART SIX

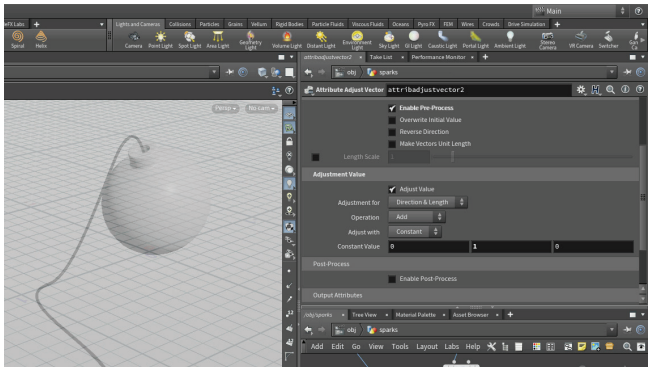
Create Particle Sparks

To create sparks, start by copying the soot particle object and make changes to the new object to generate sparks. These particles will have shorter lifespan and will be more active. The Spark Trail node will give you the look you need to add sparks to the shot. You can adjust parameters on this node to get the look you need.



01 Go to **frame 1** then navigate to the **Object** level. Hide the **Ground** object. **Alt-drag** on the *soot_trail* object to make a copy. Name the copy *sparks*.

This new object already has a popnet and can be modified to generate the spark particle simulation. In Houdini, it is often a good idea to re-use a network you already have as opposed to building everything from scratch.

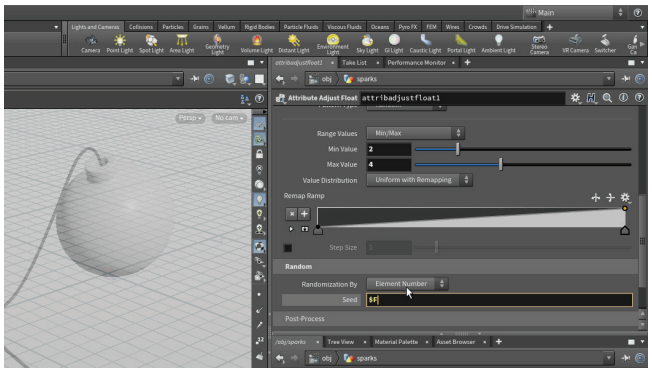


02 Dive into the *sparks* object. You will make a few changes to get the network set up to create sparks.

Delete the *atttributecreate* and *color* nodes. These are not needed for this particle network. Rename the null node to *SPARKS_OUT*.

On the *attributeadjustvector* node change **Amplitude** to **1.75.**

Add a new **Attribute Adjust Vector** node above this one. Turn on **Enable Pre-Process** and set **Constant Value** to **0, 1, 0**. Now the particles will rise up before dropping down. Press **Play**.



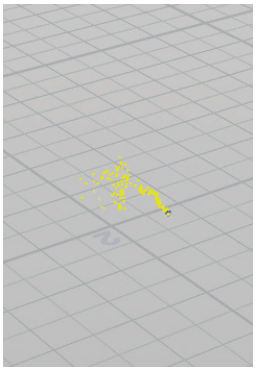
03 Got Frame 1. Add an **Attribute Adjust Float** node just before the *popnet* node. Set the following:

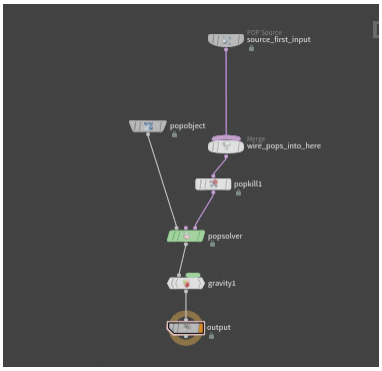
- **Attribute Name** to **life**
- **Unit Settings** to **Duration**
- **Pattern Type** to **Random.**
- **Min Value** to **2** and **Max Value** to **4**
- Under **Random**, set **Seed** to **$F**

SPARKS | PARTICLE TRAIL

The **Particle Trail SOP** takes an animated particle system and generates motion trails from its particles. These trails can be used for various effects such as sparks, fireworks, and rain.

This node also allows you to control the look of the trails. This enables you to fine-tune their look in the SOP context, without the need to render the points with large amount of motion blur.

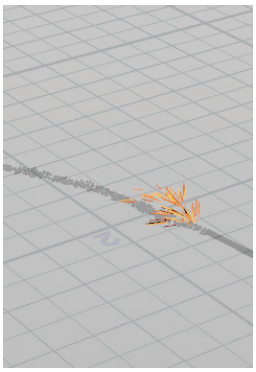


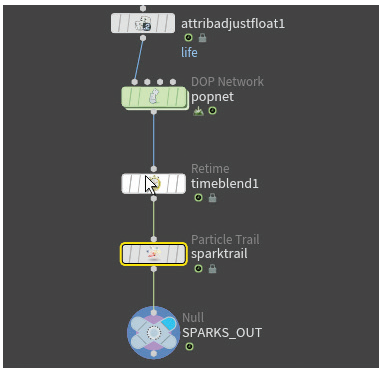


04

Dive into the *popnet* and **delete** the two *collisiondetect* nodes. You will not need to worry about collisions at this point. You will add other nodes back later when needed.

Press **Play** to see the particles being emitted.



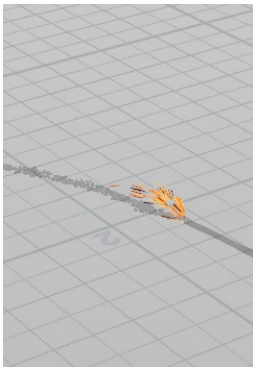


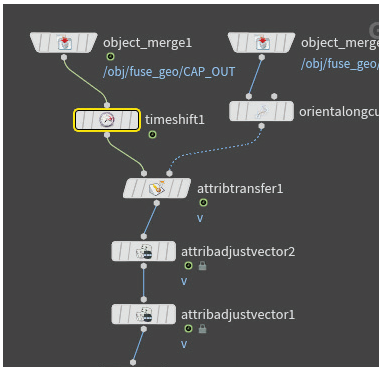
05

Go back up to the Geometry level. Go to frame 1.

In the Network view, press **tab > Spark Trail.** After a short delay, this will put down two nodes - a *timeblend* and a *sparktrail* node. Wire the *popnet* into the *timeblend* node and the *sparktrail* into the *SPARKS_OUT* null.

Press **Play**. You will see sparks but they will be lagging behind by the end of the fuse.

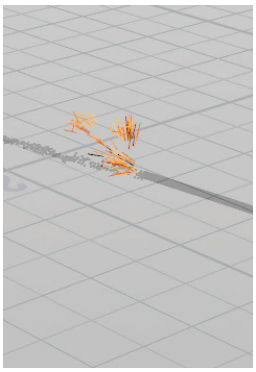


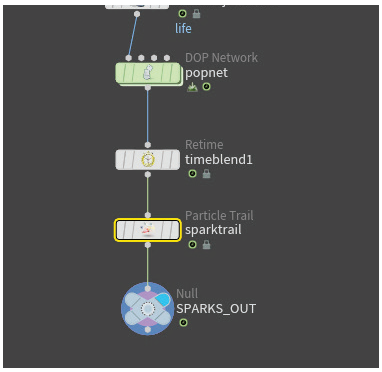


06

Add a Time Shift node after the *object_merge* which brings in the cap and the *attributetransfer* node. Click on the **Frame** parameter name to see the expression being used. Change it to `$FF + 1`

This will move the cap forward a frame but at the same time align the sparks with the end of the fuse. Press **Play** to test.



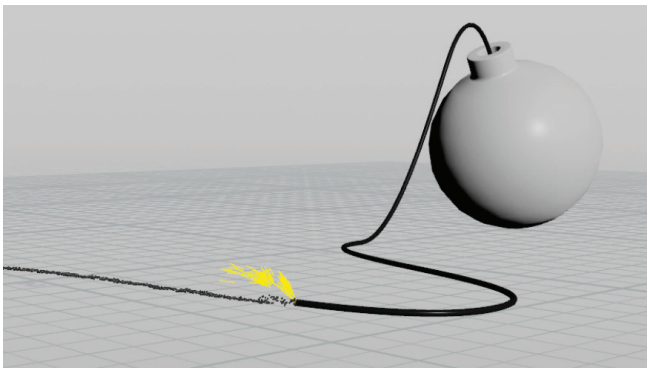


07

Go to frame 1. Select the *sparktrail* node. Click on the **Split** tab and turn **on** the **Enable Split** checkbox. Set **Percent to Split** to **40**.

Under **Shape**, set **Splits per Point** to **4**.

Press **Play** to test.



08

Go to frame 1. Jump up to the Object level.

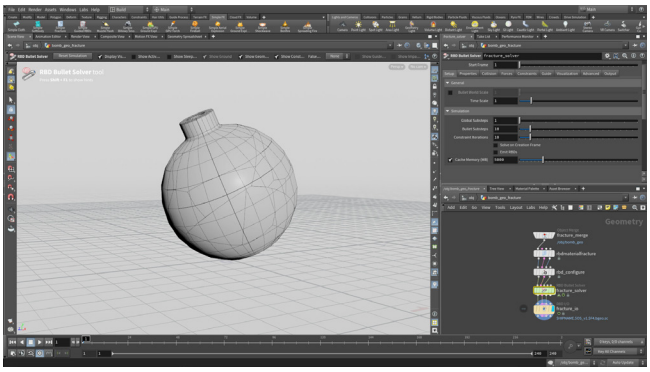
Set the camera to *cam1* and press the **Flipbook** button on the bottom of the toolbar. Click **Start** on the window that pops up. This will create an animated sequence of the scene view that you can use to evaluate the motion of your particles.

Make sure the **Realtime toggle** is on and playback the flipbook to see how the shot is progressing.

PART SEVEN

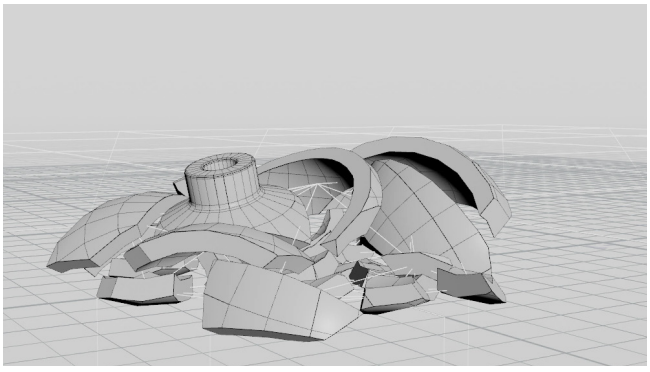
Blow up the Bomb

For the bomb geometry, a rigid body dynamics simulation will be needed. Start by fracturing the geometry then adding attributes that will create an explosion. You can then control the speed of the moving parts to art direct the look. Once the simulation is ready, you will cache out the geometry to work more efficiently as you move on to the PyroFX stage.



01 Hide the *fuse_geo*, *soot_trail*, and *sparks* objects. Press **Spacebar-G** to focus on the bomb geometry. Go to **Frame 1** in the Playbar. Press **Spacebar G** to focus on the bomb.

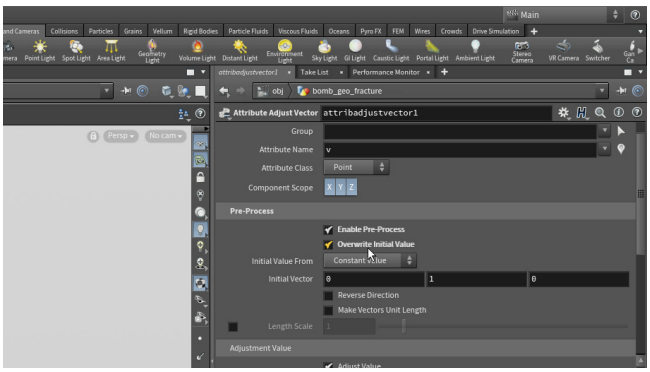
Select the *bomb_geo* object and from the **Simple FX** shelf, select **Simple Fracture**. When asked to *Select Collision Object* just press **Enter** with nothing selected. It will then take a short time to set up this network. This object merges out the bomb geometry and sets up nodes for fracturing and simulating.



02 On the *fracture_solver* node, set **Start Frame** to **200**. Now set the start frame in the **Playbar** to **200**. Click the **First Frame** button to go to frame 200. You will only need to simulate the explosion from frame 200 to 240.

On the *fracture_solver* node go to the **Collision** tab in the **Ground Collision** section, set **Ground Type** to **Ground Plane**. Click on the **Advanced** tab then in the **Constraints >Glue** section, remove the word *Glue* from the **Data Name** field.

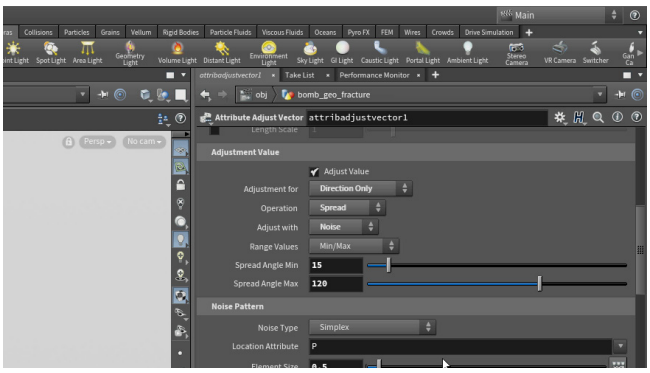
Press **Play** to watch the bomb fall apart. Now you want to add a starting velocity to explode the parts.



03 Go to **Frame 200**. **Add an Attribute Adjust Vector node and place it to the right of the other nodes. Wire the third output of the *rbd_configure* node into the *attributeadjustvector* node then wire the *attributeadjustvector* node into the third input of the *fracture_solver* node.**

Set the following on this node:

- **Turn on Enable Pre-Process**
- **Turn on Overwrite Initial Value**
- **Initial Vector to 0, 1, 0**

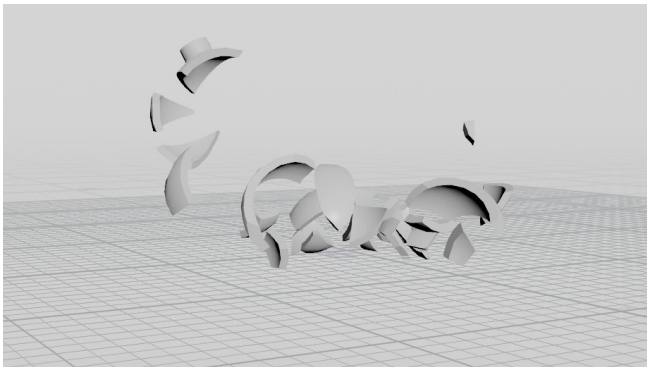


04 Under **Adjustment Value**, set the following:

- **Adjustment for** to **Direction Only**
- **Operation** to **Spread**
- **Adjust with** to **Noise**
- **Spread Angle Min** to **15**
- **Spread Angle Max** to **120**

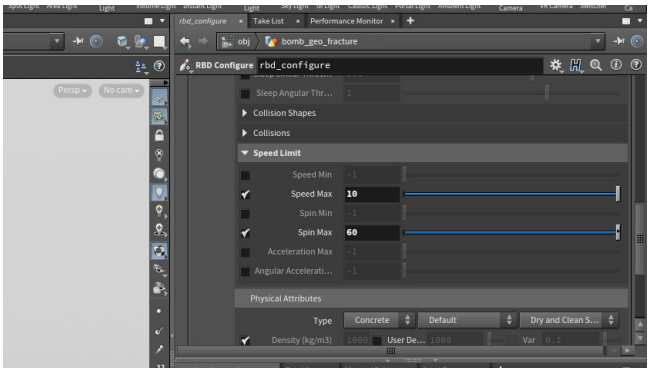
Under **Noise Pattern** set **Element Size** to **0.5**.

Under **Post-Process** turn on **Post Process** then turn on **Length Scale** and set it to **20**.



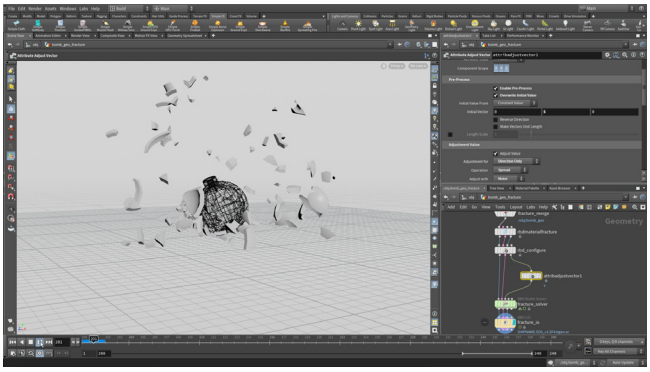
05 Press **Play**. Now you have the bomb exploding. Setting velocity attributes on the geometry feeds into the simulation which uses the initial velocity of the pieces to propel them forward.

The manipulation of attributes plays a bit role in many visual effects setups in Houdini.



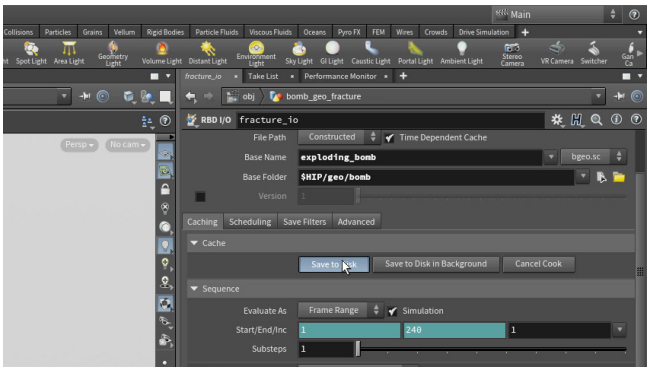
06 Select the *rbdconfigure* node. Open up the **Speed Limit** section and turn on the **Speed Max** and **Spin Max** parameters. Set **Speed Max** to **2** and **Spin Max** to **30**. Press **Play**. The simulation slows down a lot from its original speeds.

Now set **Speed Max** to **10** and **Spin Max** to **60**. Press **Play**.

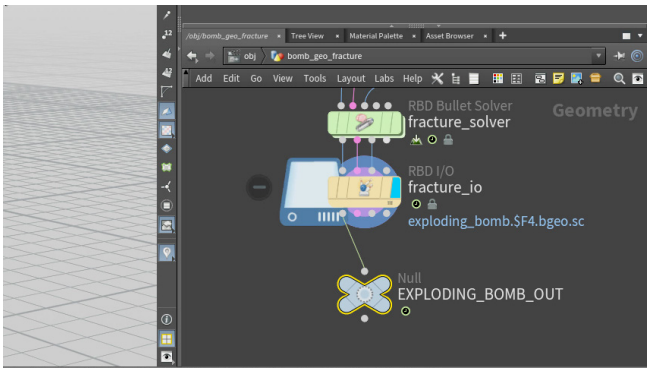


07 Select the *rbdmaterialfracture* node and under **Cell Points** set **Scatter Points** to 25. Press **Play**. Now you have a lot more pieces.

Go back to the *attributeadjustvector* node and set **Length Scale** to **50** and **Initial Vector** to **0, 5, 0**. Press **Play**.



08 Go to **frame 200**. On the *fracture_io* node, set **Base Name** to *exploding_bomb* and **Base Folder** to *$HIP/geo/bomb*. Click **Save to Disk** which will then turn on **Load From Disk**. Press **Play** to preview the cached geometry.

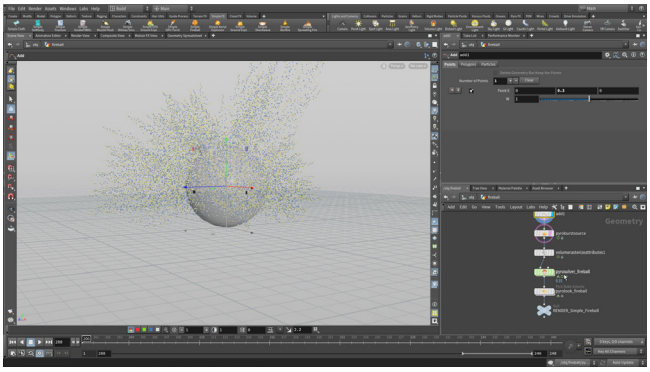


09 Add a **Null** node and wire the first output of the *fracture_io* node into it. **Rename** this node *EXPLODING_BOMB_OUT*.

PART EIGHT

Create the PyroFX Explosion

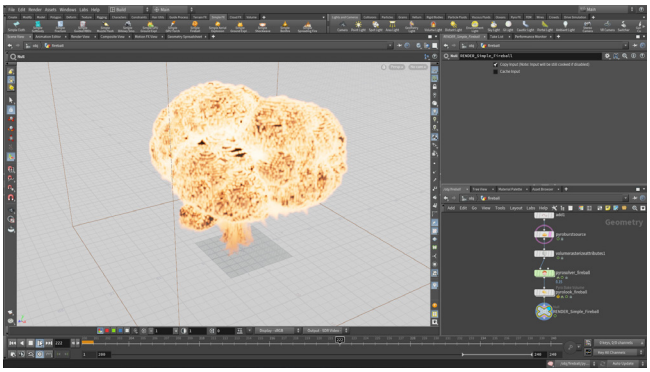
As the bomb explodes there needs to be an accompanying fireball. You will start with a Simple Fireball that works on the GPU then make changes to create a look that works for the shot. You can also incorporate the parts of the exploding bomb to push and influence the PyroFX volume in interesting ways.



01 Go to the **Object** level. From the **Simple FX** shelf, click on **Simple Fireball**. In the Scene view, press **Enter** to place it at the origin. This creates several nodes inside a *fireball* object.

Set the **Display Flag** on the *pyroburstsource* node and on the **Burst Animation** tab, set **Start Frame** to **200**. Go to **frame 200**. This node represents the initial blast of the explosion.

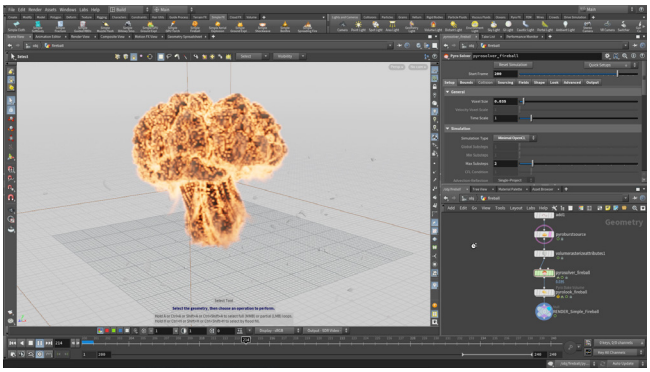
From the **Quick Setup** menu, choose **Single Input Point**. This adds a single point. Use the transform handle to raise it up to around 0.3 which is the middle of the bomb geometry.



02 On the *pyrosolver_fireball* node, change **Start Frame** to **200**. Set **Simulation Type** to **Minimal OpenCL** to simulate using your GPU.

Under the **Sourcing** tab, open the **Limit Source Range** option and set the **Frame Range** to **200, 240**. Turn off **Cycle Length.**

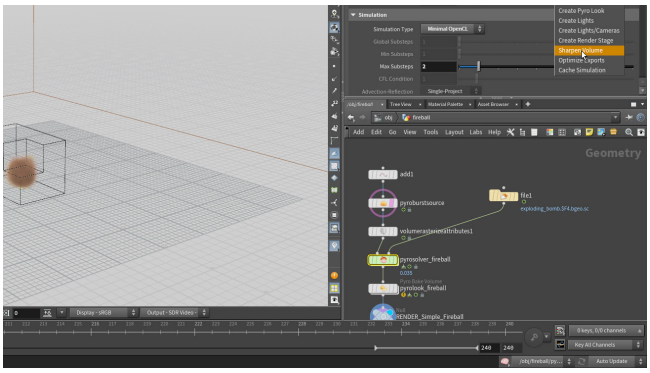
Set the **Display Flag** on the *RENDER_Simple_Fireball* null node. Zoom out to see more of the scene then press **Play** to test the simulation. The Fireball is very big. Zoom out to see the whole explosion.



03 Go to **frame 200**. On the *pyroburstsource* node go to the **Burst Shape** tab and set **Initial Size** to **0.35** and **Spread Angle** to **180**.

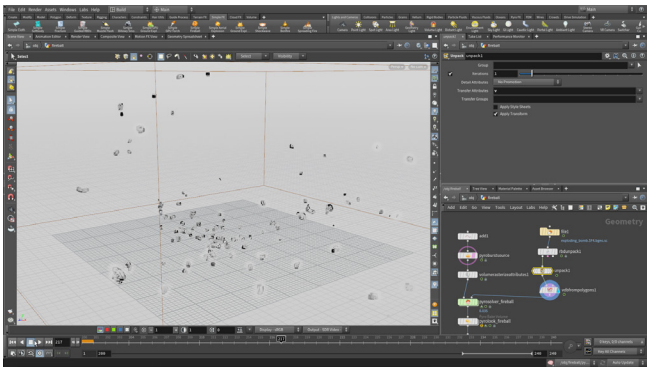
Now go to the *fireball* Pyro Solver node and click on the **Bound** tab and set **Size** to **15, 12, 12** and **Center** to **0, 4, 0**. This will make a smaller box which will fit better on the GPU. On the **Setup** tab, set the **Voxel Size** to **0.035** - this will add more detail to this smaller simulation.

Press **Play** to test the simulation. Now the explosion fits better in this scene.



04 The next step is to integrate the Pyro FX with the exploding bomb. Add a **File** node and load the *$HIP/geo/bomb/exploding_bomb.$F.bgeo.sc* geometry sequence from disk. Wire this node into the second input of the *fireball* Pyro Solver node.

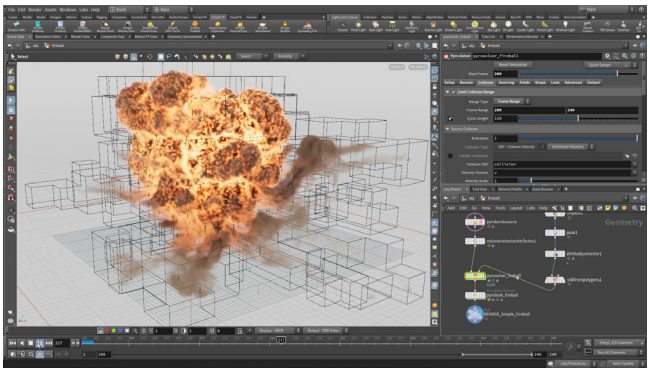
Press **Play** to test the simulation. The collision geometry isn't having an effect because it hasn't been prepared properly.



05 Go to frame **200**. Select the *fireball* node. From the **Quick Settings** menu in the top right, choose **Setup SDF Collision**. This adds a *vdbfrompolygons* node. This node gets it's **Voxel Size** from the Pyro Solver node.

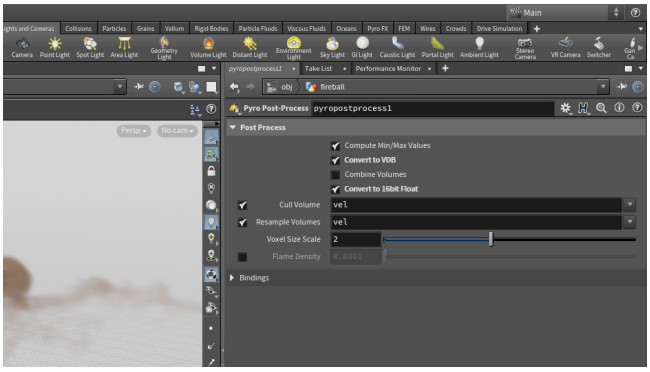
Add an **RBD Unpack** node between the *file* and *vdbfrompolygons* nodes.

Add an **Unpack** node between the *rbdunpack* and *vdbfrompolygons* nodes. On the *unpack* node, set **Transfer Velocity** to **v**. Set the **Display Flag** on *vdbfrompolygons* then press **Play** to see the collision geometry.



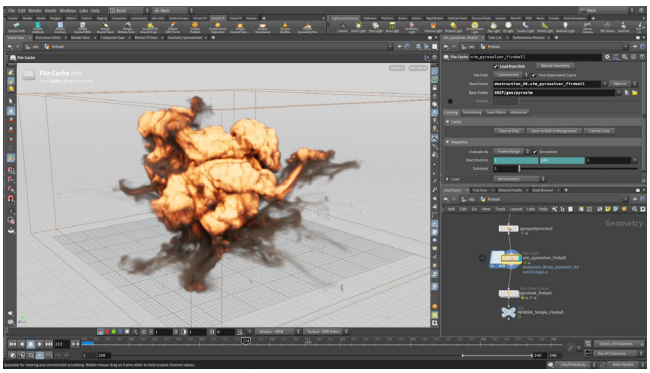
06 Between *unpack* and *vdbfrompolygons*, add a **Peak** node to make pieces bigger. Set **Distance** to **0.1**. Between the *peak* node and the *vdbfrompolygons* node, add a **Attribute Adjust Vector** node. Turn off **Adjust Value** and turn on **Enable Post Process** then turn on **Length Scale** and set it to **2**.

Set the **Display flag** back to the *fireball* node. From the **Collision** tab, open the **Limit Collision Range** section and set **Range Type** to **Frame Range** and **Frame Range** to **200, 240.** Turn off **Cycle Length.** Press **Play** to test the simulation. The exploding bomb is now colliding with and influencing the Pyro FX simulation.



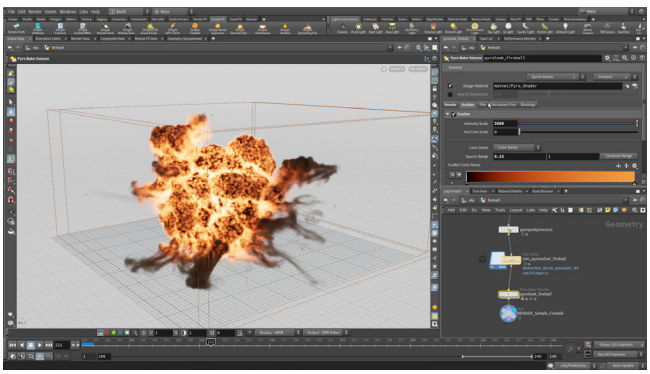
07 Between the *fireball* Pyro Solver node and the *pyrolook* node, insert a **Pyro Post Process** node. Turn on the **Convert to VDB** and the **Convert to 16bit Float** checkboxes. Next turn on the **Cull Volume** and **Resample Volumes** options and leave them both set to *vel*.

This node will make your volume more efficient and save you disk space when you cache your volume.



08 On the *fireball* Pyro Solver node click on **Quick Setups** and choose **Cache Simulation**. Move the node to the side then wire the *pyropostprocess* node into the sim_fireball cache node then wire the cache node into the *pyrolook* node. Set Base Folder to *$HIP/geo/pyrosim/* then click **Save to Disk**.

Now with the **Load from Disk** option set to on, you can scrub in the Playbar to see the PyroFX explosion.



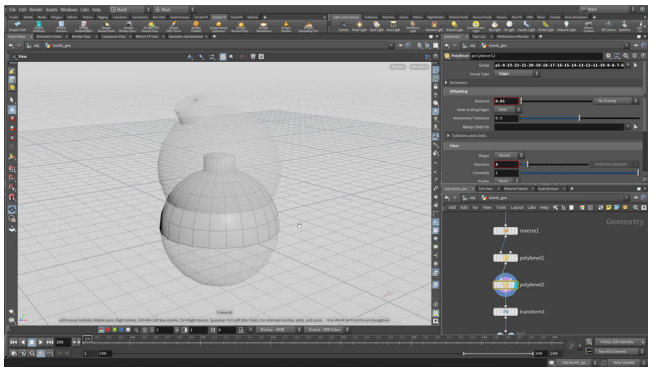
09 Set the Display Flag on the *pyrolook* node. This is the **Pyro Bake Volume** node that can be used to visualize the simulation in the Scene View. It is designed to provide a similar interface to the Pyro Shader you will use later when rendering.

Under the **Smoke** tab, darken the **Smoke Color** then under the **Scatter** tab set **Intensity Scale** to **2500**. Make other adjustments to get a look that you like for your simulation.

PART NINE

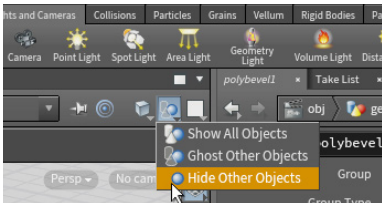
Export the Geometry to USD

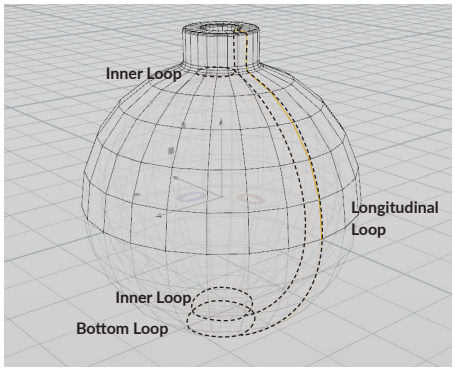
To set up the shot for rendering, you need to export the geometry to USD files that can be referenced into the Solaris context. While you could import the geometry directly, having it cached out as USD will allow you to lock down your sequences then focus on lighting and rendering in Solaris. For some of these objects, you will add UVs before exporting to prep for texturing.

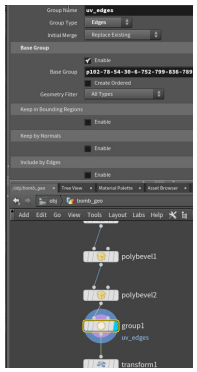


01 Go to the Object level and hide the *fireball* object. Double click on the *bomb_geo* and set the **Display Flag** on the second *polybevel* node. This shows you the bomb geometry before it was rotated into place.

Set the **Display** to **Hide Other Objects** to stay focused on the contents of this network.



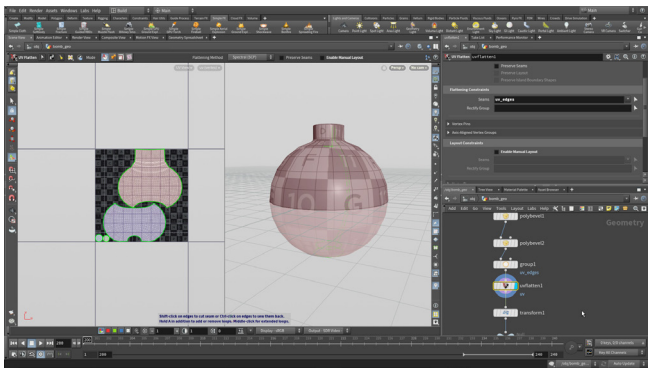




02 Go to the **Select** tool then **press 3** to select **Edges**. **Press w** to go into wireframe mode and then press **Shift** and **double click** to select part of the loop at the bottom. Press **Shift** and **double click** to select all four parts. Repeat for the inner loop.

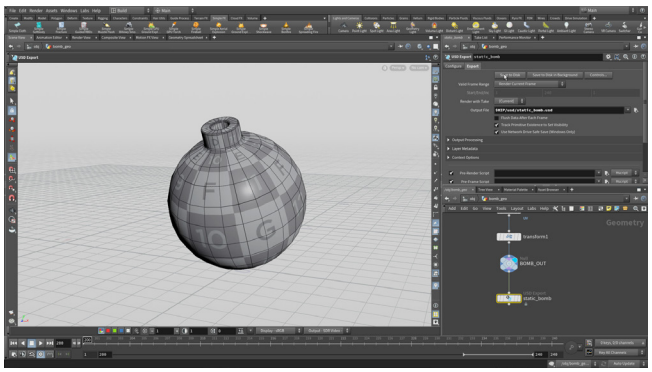
Now go to the loop on the top of the inner sphere and press **Shift** and **double click** to select. Now pick the latitudinal line that aligns with the X axis and press **Shift** and **double click** to select.

In the Scene View, press **tab > Group**. Change the **Group Name** to *uv_edges*.



03 In the Scene View press **tab > UV Flatten** which adds a *uvflatten* node after the *group* node. Set **Seams** to *uv_edges* and under **Layout Constraints**, turn off **Enable Manual Layout**. This provides a good looking UV layout that will be perfect for texturing the bomb.

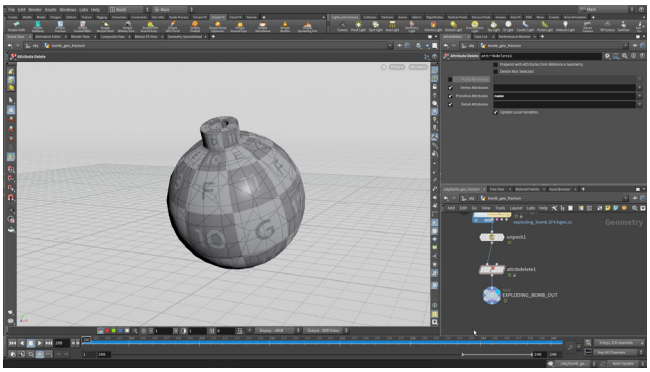
Set the **Display Flag** on the *BOMB_OUT* null node.



04 Add a **USD Export** node to the end of the chain. **Rename** it *static_bomb*. Set the following

- **Valid Frame Range** to **Render Current Frame**
- **Output File** to *$HIP/usd/static_bomb.usd*

Press **Save to Disk**.

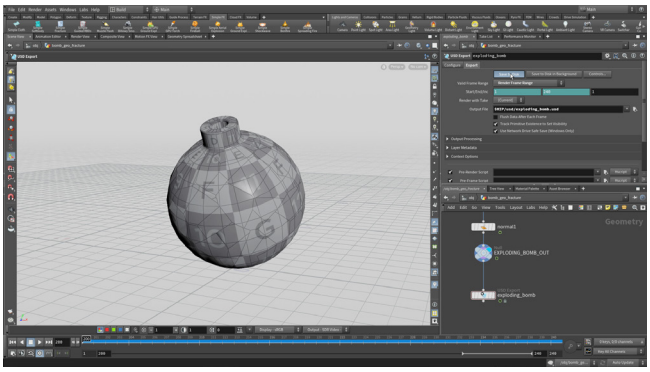


05

Go back to the Object level then double-click on the *bomb_geo_fracture* network. Select the *fracture_io* node and press **Save to Disk** to cache out the geometry with the new UVs. You won't see them yet because the geometry is packed.

After the *fracture_io* node and before the *EXPLODING_BOMB_OUT* node, add an **Unpack** node then an **Attribute Delete** node. On the *attributedelete* node, enter *name* next to **Primitive Attributes**. Turn off the **Point Attributes** section.

This ensures that the sequence comes into Solaris as a single mesh. The name attribute would break the sequence into individual parts.



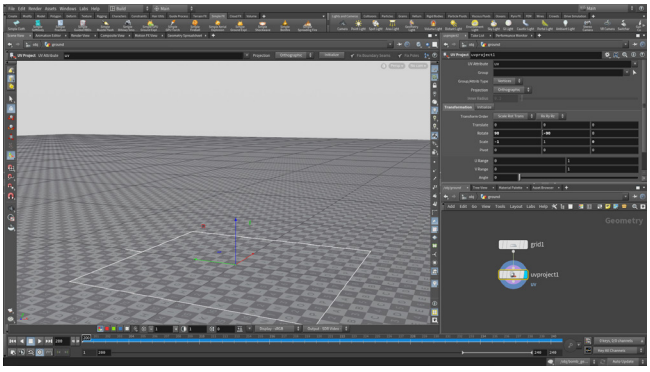
06

After the *attributedelete* node, add a **Normal** node. This will help the bomb geometry display properly in Solaris.

Add a **USD Export** node to the end of the chain. **Rename** it *exploding_bomb*. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/exploding_bomb.usd*

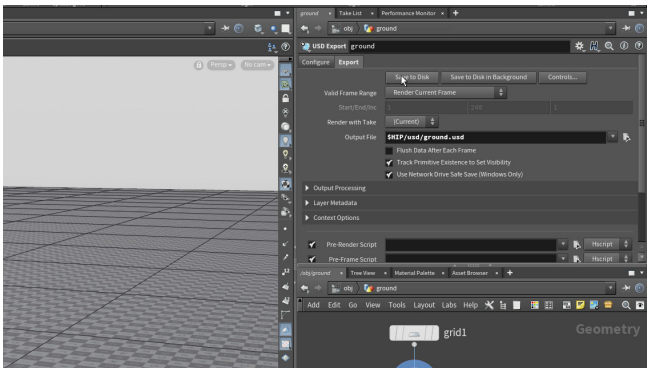
Press **Save to Disk.**



07

Go back to the Object level then **double-click** on the *ground* network. After the *grid* node, add a **UV Project** node and set its **Display Flag**. Go to the **Initialize** tab and press the **Initialize** button. On the **Transformation** tab, change **Scale X** to **-1**, **Scale Y** to **1** and **Rotate Y** to **-90**.

This will allow the texture to repeat on the ground surface rather than create one large texture,



08

Add a **USD Export** node to the end of the chain. **Rename** it *ground*. Set the following

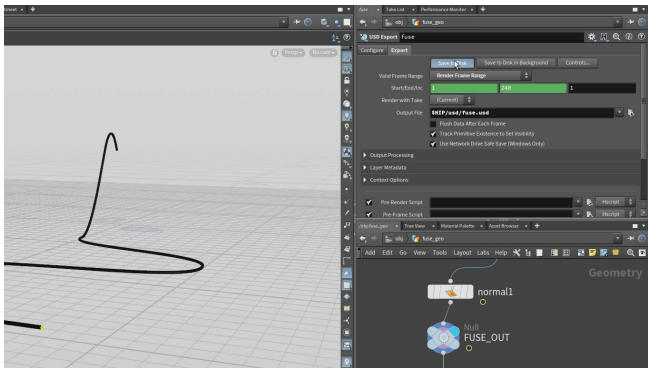
- Leave **Valid Frame Range** set to **Render Current Frame**
- **Output File** to *$HIP/usd/ground.usd*

Hit **Save to Disk.**

USD and SOLARIS

To support the look development stage of this project, layout, lookdev and lighting workflows are set up in the **Solaris** context. This is represented by **LOP networks**. The **USD** caches you are creating here will go into the Solaris context.

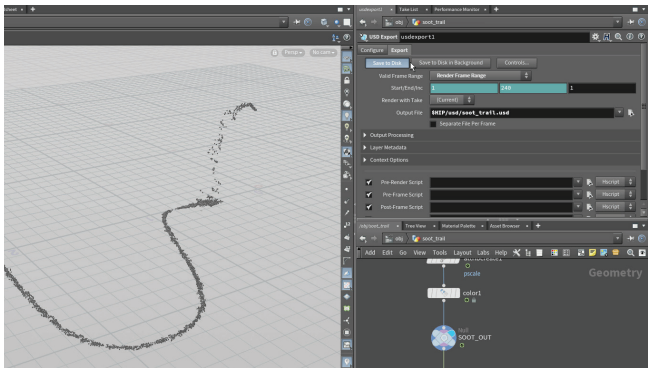
You will reference the USD caches into the LOP network using this scene file but in a larger pipeline another option would be to start a new scene file and import the USD files in a fresh scene. This would let you focus on lighting and rendering your shot but would make it harder to go back and tweak the geometry and simulations.



09 Go back to the Object level then **double-click** on the *fuse* network. Add a **USD Export** node to the end of the chain. **Rename** it *fuse*. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/fuse.usd*

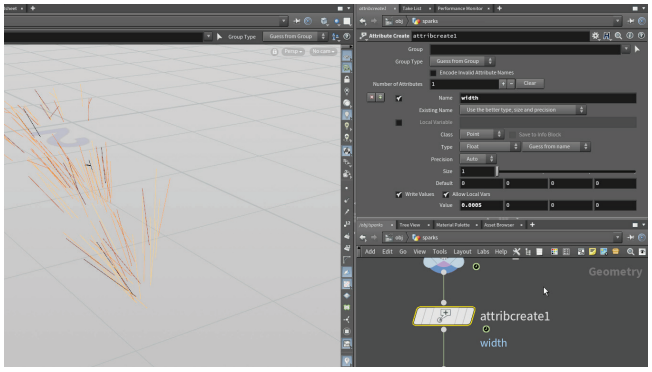
Hit **Save to Disk.**



10 Go back to the Object level then **double-click** on the *soot* network. Add a **USD Export** node to the end of the chain. **Rename** it *soot_trail*. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/soot_trail.usd*

Hit **Save to Disk.**



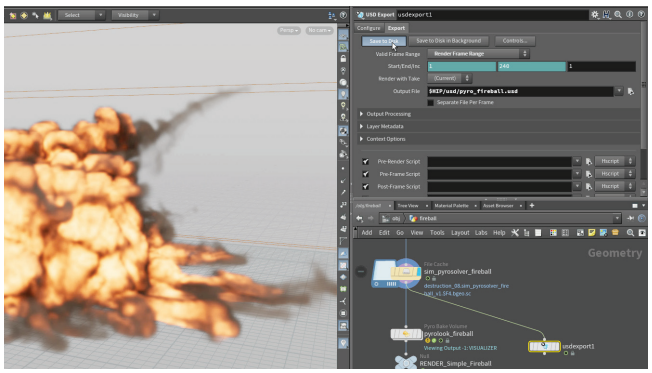
11 Go back to the Object level then **double-click** on the *sparks* network.

Add an **Attribute Create** node to the end of the chain. Set the **Name** to *width* and the **Value** to **0.0005**. This will determine the look of the sparks when they are rendered.

Add a **USD Export** node to the end of the chain. **Rename** it *sparks*. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/sparks.usd*

Hit **Save to Disk.**



12 Go back to the Object level then **double-click** on the *fireball* network. From the *sim_fireball* node, branch off a **USD Export** node. Be sure to bypass the *pyrolook* node. **Rename** it *pyro_fireball*. Set the following

- **Valid Frame Range** to **Render Frame Range**
- **Output File** to *$HIP/usd/pyro_fireball.usd*

Hit **Save to Disk.**

Scene Import

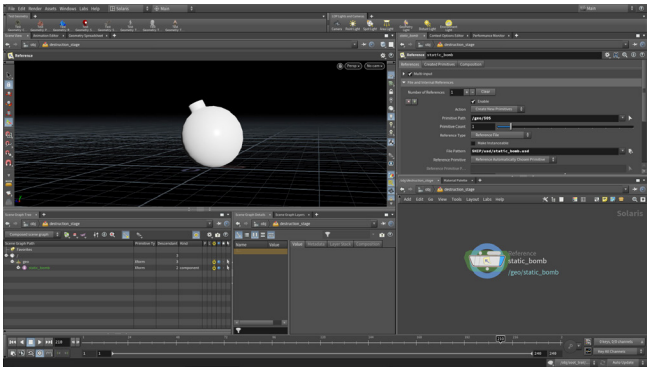
Another way to get your geometry and simulations into Solaris is using the **Scene Import LOP**. This creates a direct connection between the geometry and objects you are working with and the LOP network. This approach would require cache LOPs to support motion blur which is an extra step not needed if you are referencing in USD files.

This tool is going to be used to bring the animated camera from the object level into the Solaris context.

PART TEN

Set up the Shot in Solaris

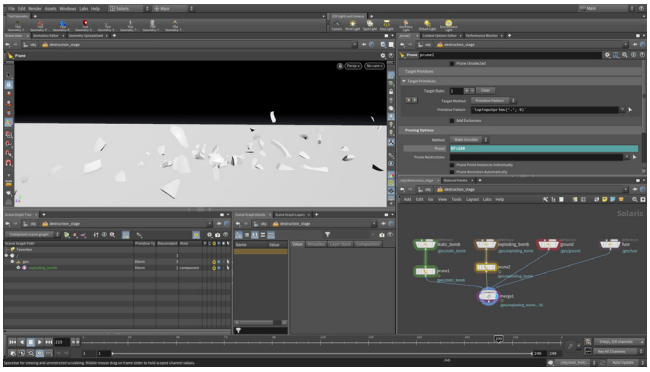
Learn how to reference all of the USD files into Solaris then import the camera from the object level. Apply materials to all the elements and start rendering with Karma to evaluate the results. Learn how to add a key light and prepare render settings to explore the final look of the shot.



01 Go back to the Object level. Press **tab > LOP Network** to create a subnetwork to use to set up your shot. Name it *destruction_stage*. **Double click** on the node to dive into it.

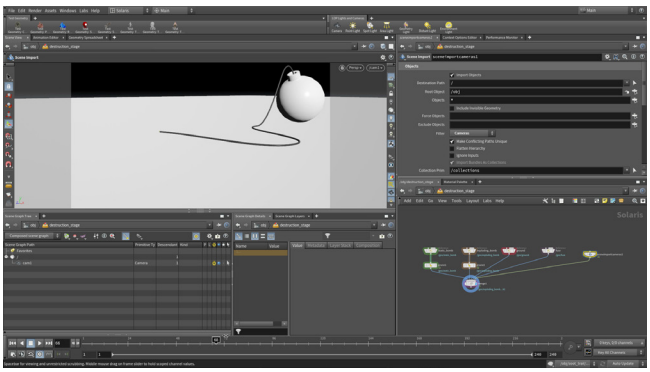
Change the desktop to **Solaris**. Make sure you see *obj > destruction_stage* in the Scene view's path bar. **Press D** over the scene view and go to the **Background** tab and set **Color Scheme** to **Dark**.

In the Network View press **tab > Reference** then click to add a *reference* node. Next to **Reference Pattern**, click on the **File Chooser** and find the *static_bomb.usd* file. **Rename** the node to *static_bomb*. Set the **Primitive Path** to */geo/$OS*.



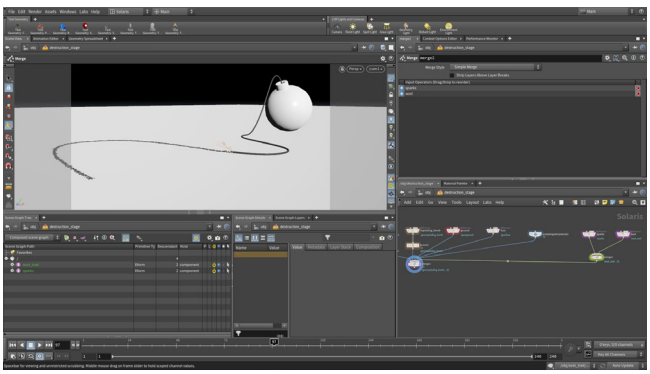
02 **Alt-drag** on this node to make three copies. Name them *exploding_bomb*, *ground* and *fuse* then point the **File Chooser** parameter to these USD files.

Feed these into a **Merge** node and set its **Display Flag**. After the *static_bomb* node, add a **Prune LOP**. Set the **Prune** parameter to *$F >200*. After the *expoding_bomb* node, add another **Prune LOP**. Set the **Prune** parameter to *$F <199*. Now when you scrub in the Playbar, the bomb will switch between the static and exploding bomb at frame 200.



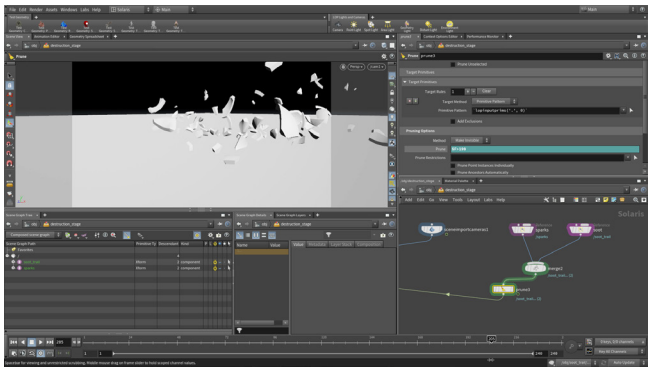
03 Press **tab > Scene Import (Cameras)**. Place down this node and wire it into the *merge* node. Set the **Destination Path** to */cam/*. This adds any camera found at the object level into the Solaris context.

Go to the **camera** menu choose *cam1* to look through this animated camera. Make sure you have the merge node selected. Scrub in the Playbar to see the fuse and the exploding bomb animate and then explode through the lens of the camera you set up at the object level. If you were to make changes to that camera it would be reflected here in Solaris.

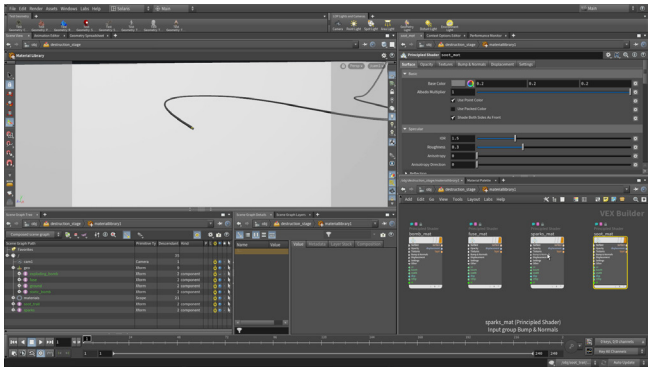


04 In the Network View press **tab > Reference** then click to add a *reference* node. Next to **Reference Pattern**, click on the **File Chooser** and find the *sparks.usd* file. **Rename** the node to *sparks*. Set the **Primitive Path** to */fx/$OS*.

Alt -drag on this node to make a copy. Name it **soot** then point the **File Chooser** and find the *soot_trail.usd* file. Add a **Merge** node to bring these files together then feed them into the main **Merge** node and set it's **Display Flag**. If you scrub in time, you will see the sparks and the soot persist after they were killed. The USD file holds onto the particles right until frame 240.

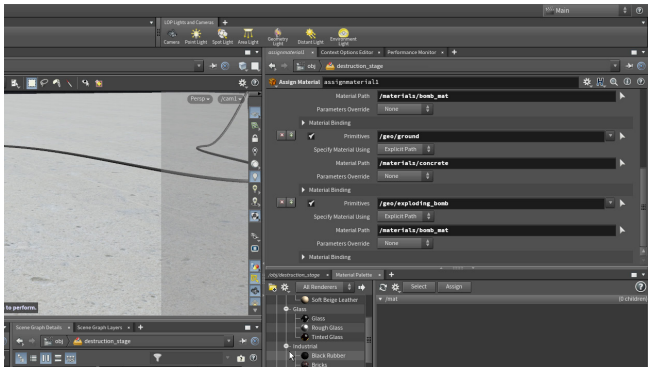


05 After the effects *merge* node, add a **Prune LOP**. Set its **Display Flag**. Set the **Prune** parameter to **$F >199**. Now the soot and sparks disappear at this point.



06 Add a **Material Library** node after the *merge* node. Go to the **Material Palette** and drag a **Principled Shader** and a **Concrete Shader** into */stage/materiallibrary*.

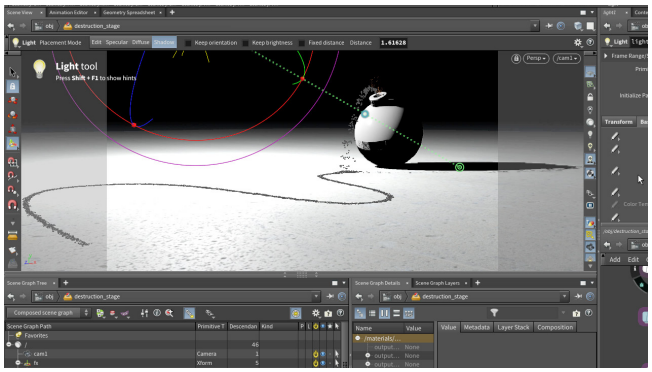
Go to the Network view and rename the principled shader to *bomb_mat*. **Alt drag** to make three copies of the principled shader and call them *fuse_mat*, *sparks_mat* and *soot_mat*.



07 Go back up one level and add an **Assign Material** node to the end of the chain. From the Scene Graph, drag *static_bomb* and *exploding_bomb* to the **Primitives** field.

Now click on the arrow next to **Material Path** and select the *bomb_mat*. Press the **plus sign** to add more sections and repeat these steps to assign materials to the *fuse*, *sparks* and *soot*.

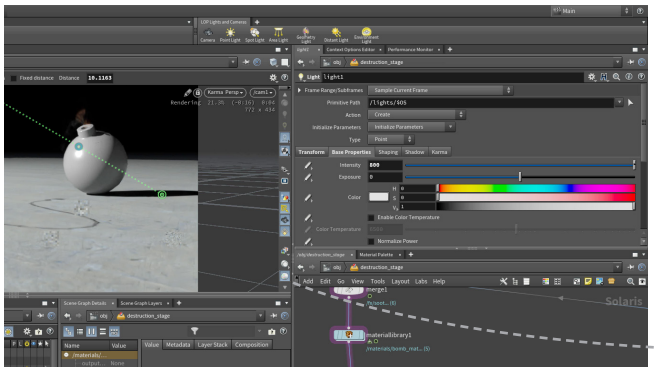
Assign the *concrete* Material to the *ground*.



08 Go to around frame 180. Click on the **Light** tool in the **LOP Lights and Cameras** shelf. This adds a light to your shot and has you looking through it.

Go to the **Base Properties** tab and set **Intensity** to **50**.

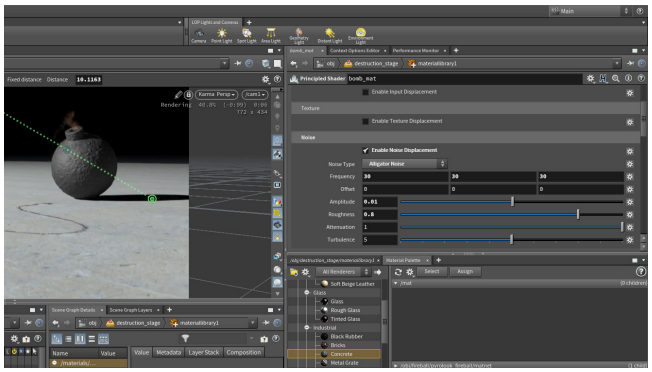
In the Scene View, click on the **Shadow** button and **click** on the surface of the bomb then **Shift click** behind it to create a shadow.



09 In the Scene view, select **Karma** from the **Persp** menu. Turn off the **Reference Plane**.

Go back into the **Material Palette** and select the *concrete* shader. In the **Texture** section, set **Effect Scale** to **0.01**.

Turn on the **Denoiser** in the side bar. This will use your graphics card (nVidia cards only) to more quickly resolve the noise in your rendering.



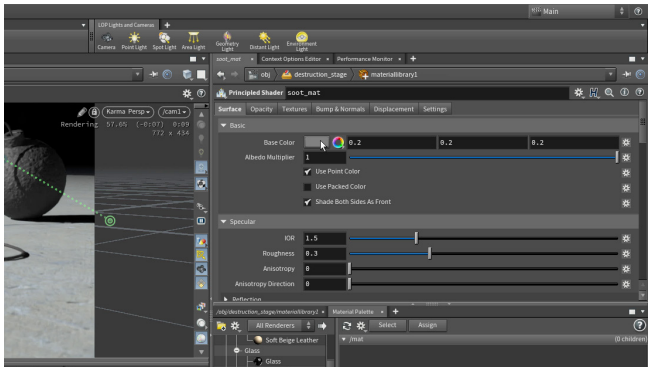
10

Select the *bomb_mat* shader. Under **Surface**, set the following:

- **Base Color** to **Black [0, 0, 0]**
- **Roughness** to **0.7**

Under **Displacement**, turn on **Enable Noise Displacement** and set the following:

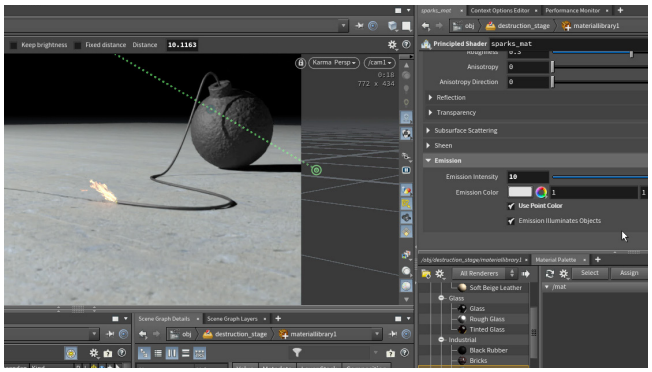
- **Noise Type** to **Alligator Noise**
- **Frequency** to **30, 30, 30**
- **Amplitude** to **0.01**
- **Roughness** to **0.8**



11

For the *fuse_mat* and *soot_mat* shaders, set the **Base Color** to a **Dark Grey**.

On the *fuse_mat* shader turn off **Use Point Color** to allow the Base Color to control the look.

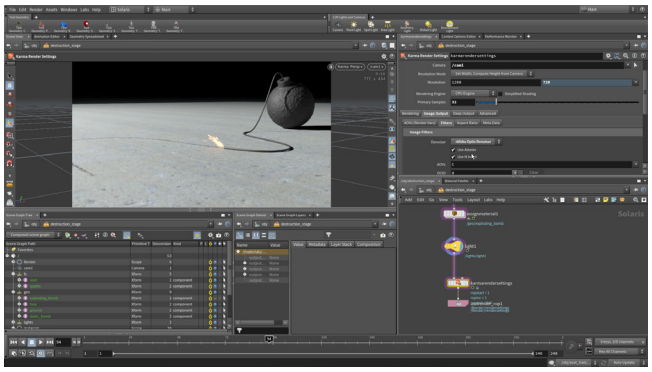


12

Select the *sparks_mat* shader and under **Emission** set the following:

- **Emission Color** to **1, 1, 1 [white]**
- **Emission Intensity** to **10**
- Turn on **Use Point Color**

This will make the sparks shine brighter and even create a bit of illumination on the ground surface.



13

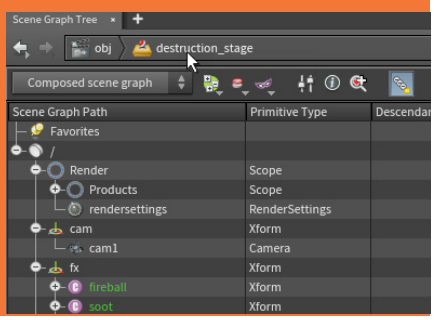
Go back to the Stage level. In the **Network View**, press **tab > Karma** to add a **Karma Render Settings** and **USD Render ROP** node. Wire them into the end of the chain. Select the *karmarendersettings* node and set **Primary Samples** to **32**. On the **Image Output > Filters** tab set **Denoiser** to **nvidia Optix Denoiser** to turn the denoiser back on.

On the **Advanced** tab, go to the **Sampling** section and set **Convergence Mode** to **Path Traced**.

RENDER SETTINGS

The **Karma Render Settings** node adds render settings that become part of the **Scene Graph** and will be used to render to the viewport and to disk.

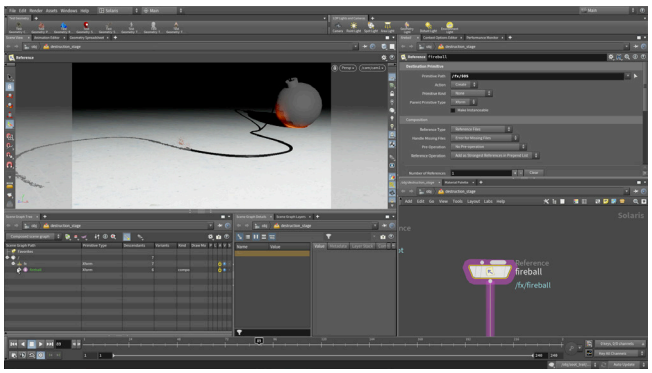
Before you added these settings the viewport settings were being used when you rendered in the viewport. If you have Karma rendering set up in the Scene view then **Press D** to see these settings. If you have render settings in your Scene Graph then they will override the viewport settings.



PART ELEVEN

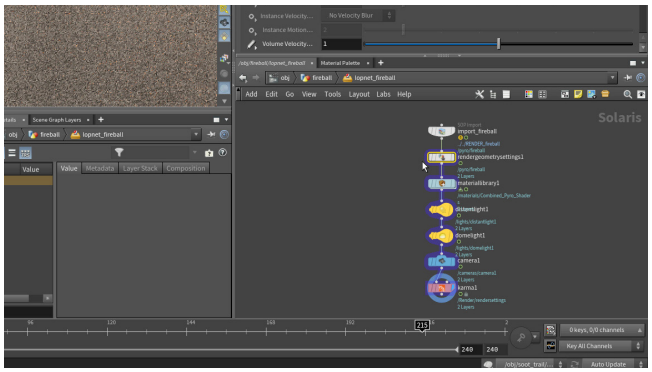
Render the PyroFX

To complete the shot, add the fireball USD file then assign the proper material. Next, you will set up another camera to create a wide angle shot of the explosion and then render out the two sequences to achieve the final sequence. You can then preview the results using the Mplay image viewer.



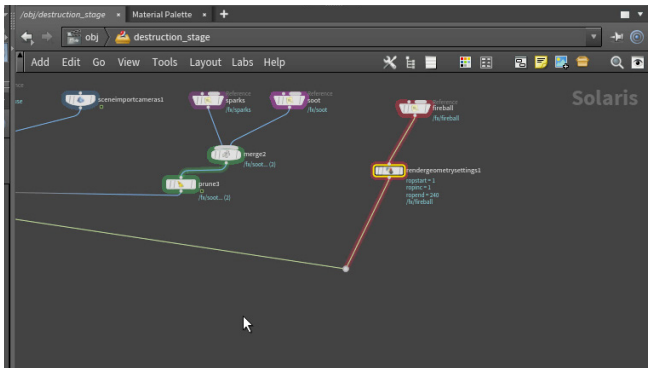
01 Set the Persp view menu back to **Houdini GL**. In the Network View press **tab > Reference** then click to add a *reference* node. Next to **Reference File**, click on the **File Chooser** and find the *pyrofx_fireball.usd* file. **Rename** the node to *fireball*. Set the **Primitive Path** to */fx/$OS*.

Feed this node into the original *merge* node. **Alt click** on the connecting wire to add a dot and move the dot to the lower right. With the **Display Flag** on the *Karma* node, scrub to around **frame 204** to see the explosion.



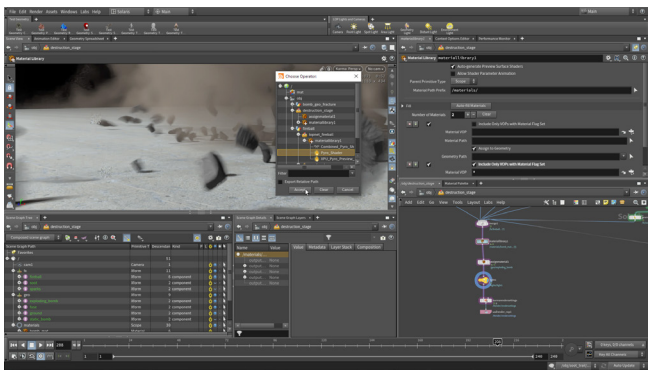
02 Navigate back to the Object level then into the *fireball* object. Select the *pyrosolver_fireball* node. From the **Quick Setups** menu, select **Create Render Stage**. This adds *lopnet_fireball* into this network. You can dive into it to see the suggested setup. You could use this to render the fireball on its own but you need it as part of our existing LOP network.

Select the *rendergeometrysettings* node and press **Cmd-C** to copy it.



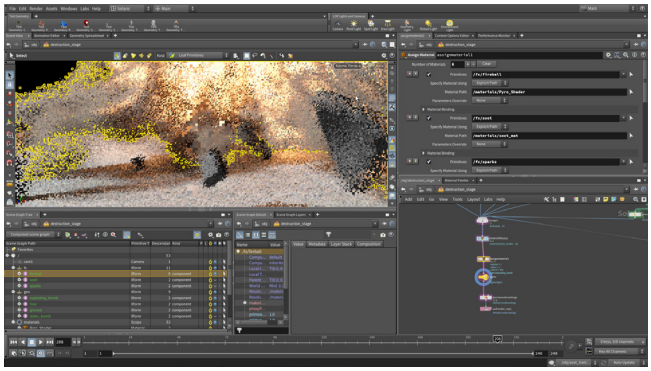
03 Navigate back to the *destruction_stage* LOP network and press **Cmd-V** to paste it into the network. Wire it just under the *fireball* reference node.

This node does two things. First it sets up velocity motion blur on the fireball then it uses the volume to help light the shot.



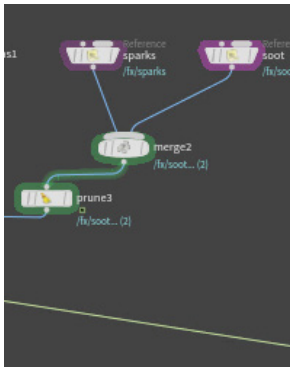
04 Select the *materiallibrary* node and click the plus sign next to **Number of Materials**. Click on the Operator chooser button next to **Material VOP** on the new listing and navigate into the *fireball* object then into the *lopnet_fireball* then into the *matnet* to select the **Pyro_Shader**. Click **Accept**.

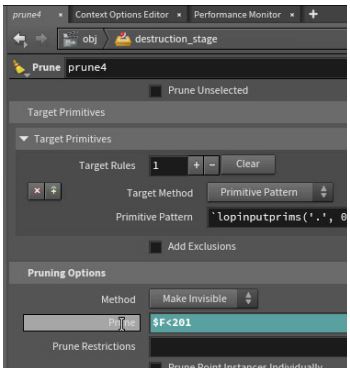
Even though this material is in a different LOP network, it can be referenced from its location to this *materiallibrary* node.



05 On the *assignmaterial* node, add another material listing and from the Scene Graph drag */fx/fireball* to the **Primitives** section then click on the **Material Path** arrow and choose **Pyro_Shader.**

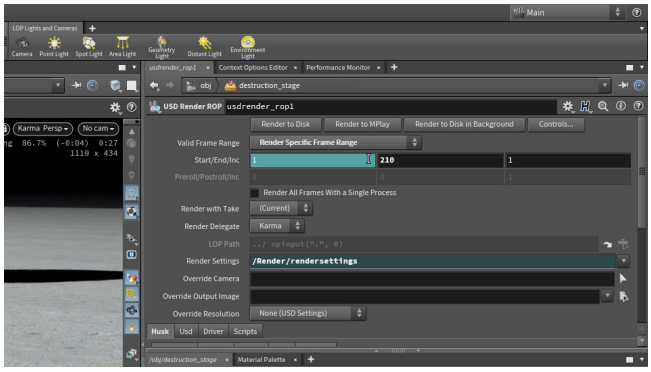
Set the Persp view menu back to **Karma**.





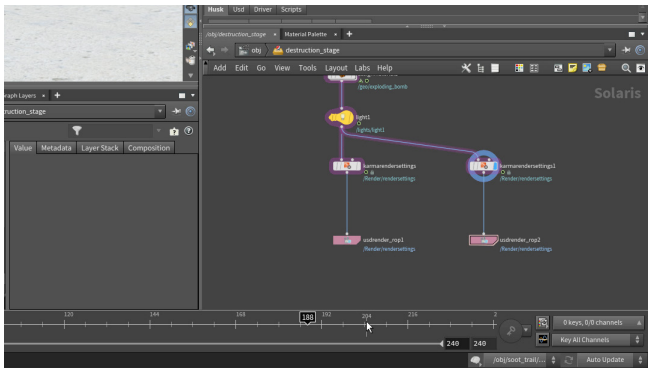
06 Add **Prune** node to the network and wire it just under the *rendergeometrysettings* node. Under Pruning Options, set **Prune** to **$F<201**.

The fireball at frame 200 was poking through the bomb geometry and this will delay the explosion for one extra frame.

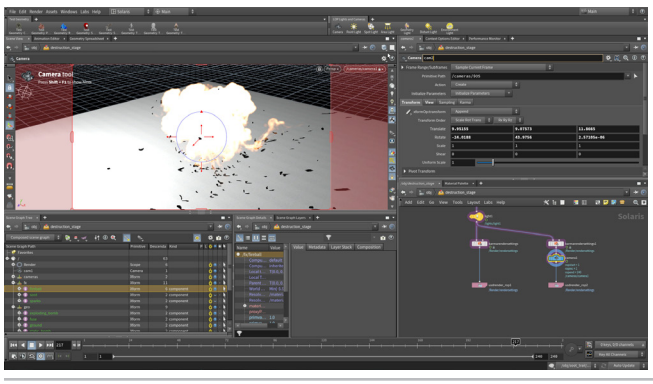


07 On the Karma node, set **Valid Frame Range** to **Render Frame Range**. **RMB-click** on the **End** value which shows **240** and select **Delete Channels**. Change the **End** value to **210**.

You will use the animated camera for the first 210 frames then cut to a different camera for the last 30.

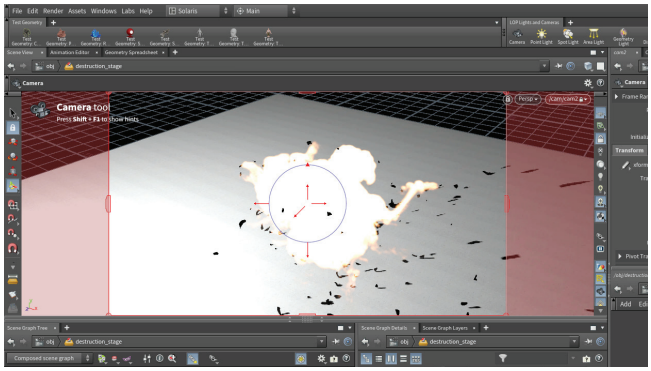


08 Set the Persp view menu back to **Houdini GL**. In the Network view, **Alt-drag** on the *karmarendersettings and usdrender_rop* nodes to the right. Set its **Display Flag**.

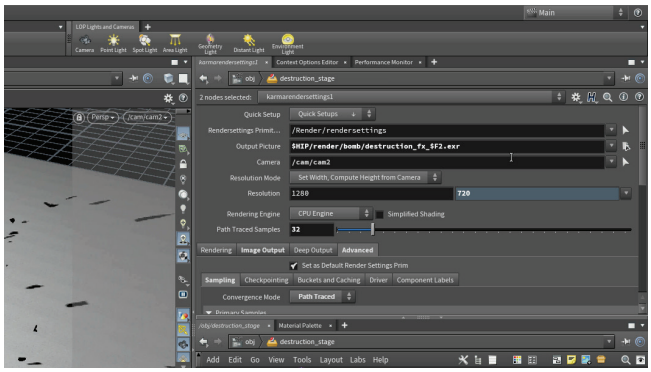


09 **Tumble** in the Scene view to get a new camera angle that is looking down on the explosion from above. Press the **Ctrl** key and click on the **Camera** tool in the **LOP Lights and Cameras** shelf.

Set its **Primitive Path** to */cam/$OS* and its **name** to *cam2*.



10 **Lock the Camera to View** and tweak the viewpoint to get the shot you are looking for. Check it at various points between frame 210 and 240. Turn the **Lock the Camera to View** option off when you are ready.



11 Move the *cam2* node up above the *karmarendersettings2* node. On the *usdrender_rop2* node, change the **Start** and **End** to **211** and **240** then change the camera to point to */cameras/cam2*.

Select the first *karmarendersettings* node and make sure that its camera is set to */cam/cam1*. Otherwise the node will not render since the default *camera1* isn't in our scene.

Select both *usdrender_rop* nodes, and change the Output picture to: *$HIP/render/bomb/destruction_fx_$F2.exr*

12 Select the *usdrender_rop1* node. Click on the **Render to Disk** button. Repeat for usdrender_rop2.

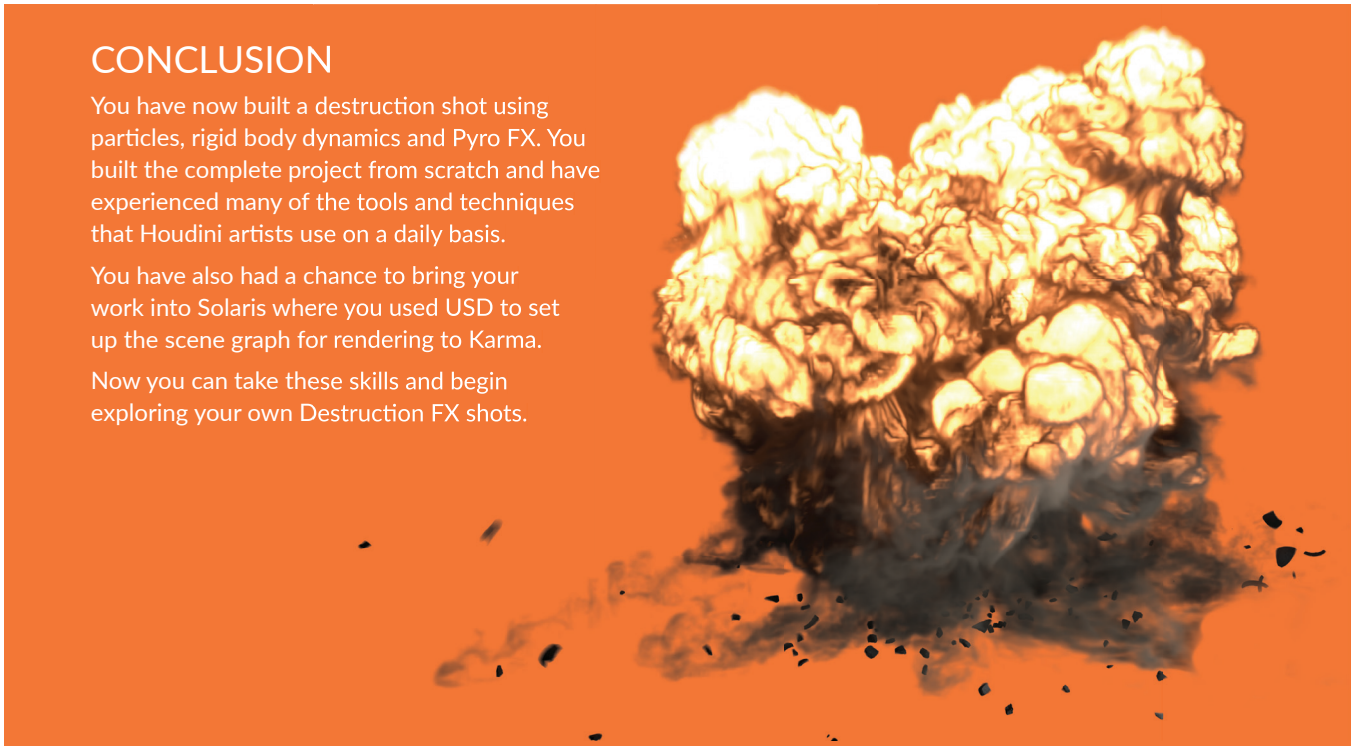
Go to the **Render** menu and select **MPlay > Load Desk Files**. Go to the *render/bomb* directory and select the image sequence then click **Load**. This will play the images as a single animation.

CONCLUSION

You have now built a destruction shot using particles, rigid body dynamics and Pyro FX. You built the complete project from scratch and have experienced many of the tools and techniques that Houdini artists use on a daily basis.

You have also had a chance to bring your work into Solaris where you used USD to set up the scene graph for rendering to Karma.

Now you can take these skills and begin exploring your own Destruction FX shots.



HOUDINI FOUNDATIONS

TERRAIN GENERATION

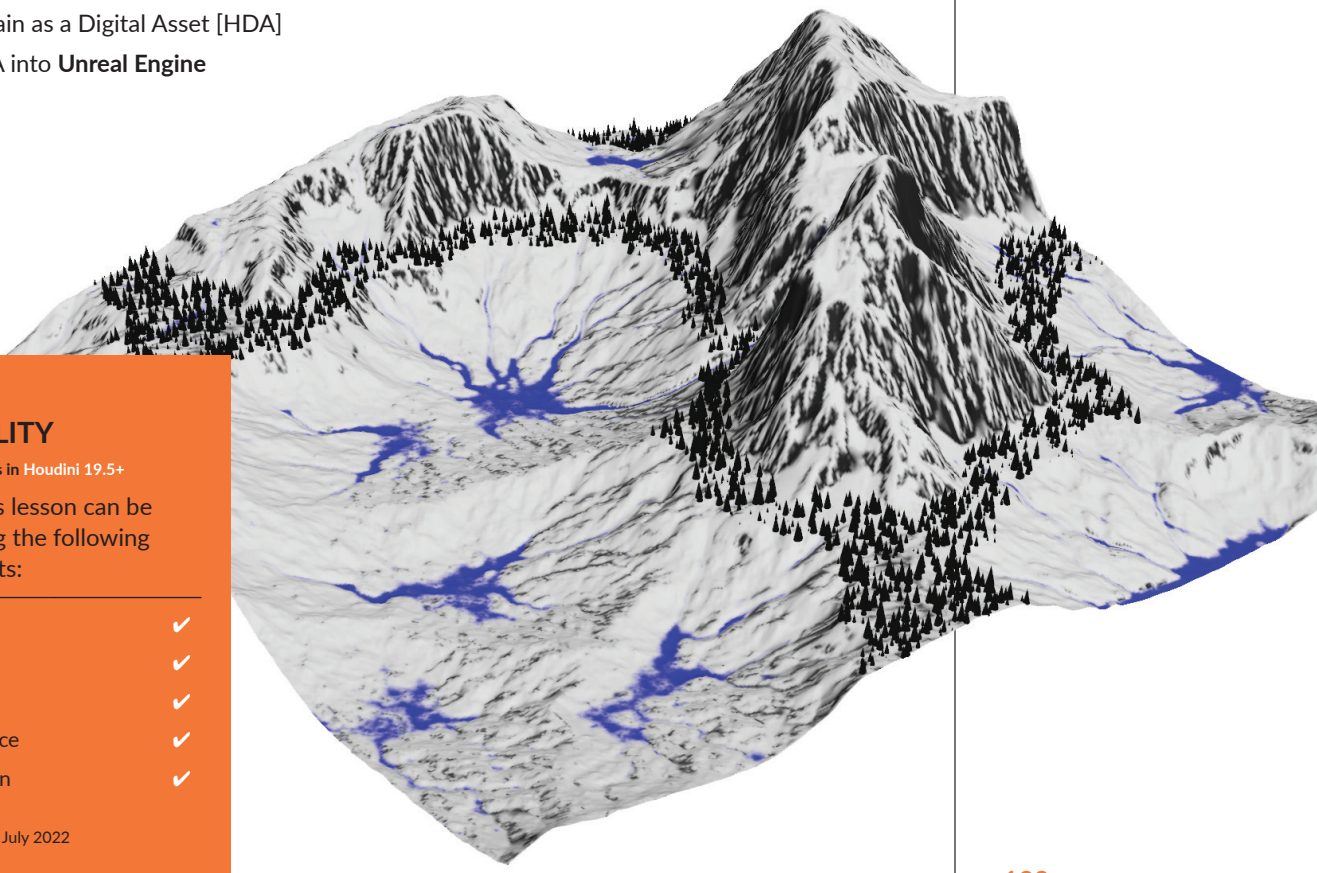
Houdini includes a dedicated toolset for generating and shaping terrains. These tools represent terrain using 2D volumes, called **heightfields,** where each voxel contains the height of the terrain at a particular grid point. The Houdini viewport lets you visualize 2D heightfields as 3D surfaces. You can also set up mask fields that can be used to focus your edits to specific parts of the terrain. In this lesson, you will build up terrains using patterns, noise and erosion then export the results for use in a game engine.

LESSON GOAL

Create a landscape using the Heightfield tools in Houdini and bring it into Unreal Engine or Unity.

WHAT YOU WILL LEARN

- How to create a Terrain using **heightfields**
- Add **patterns**, **noise** and **distortions**
- Create **Masks** using terrain features
- How to create **Scatter Points** on heightfield
- How to set up Instancing using **Terrain Scattering**
- **Export** the Terrain as a Digital Asset [HDA]
- Import the HDA into **Unreal Engine**



LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

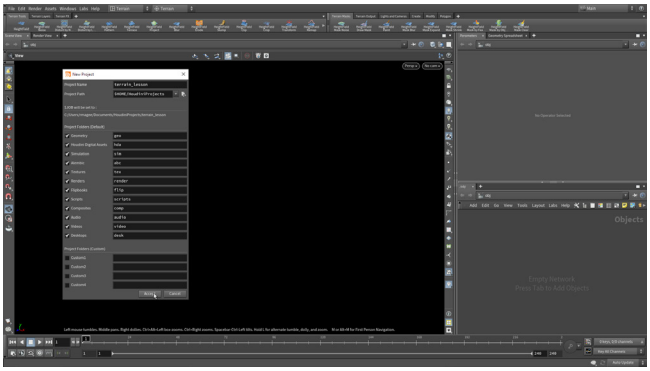
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 2.0 | July 2022

PART ONE

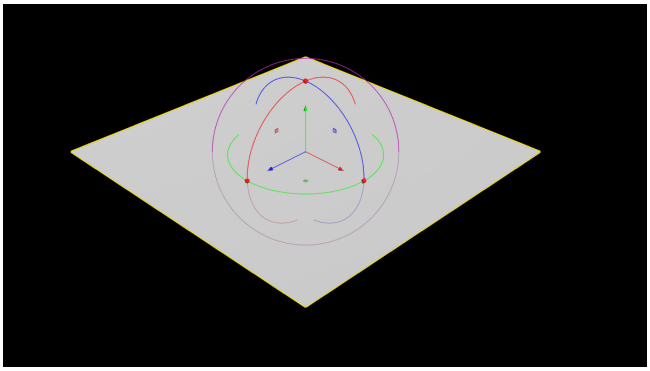
Shape the Terrain using Heightfields

To create terrains in Houdini, you will work with heightfields. You will start with a blank heightfield then add some noise and distortion to define the basic look of the landscape. As you work, you can tweak parameter values on the nodes while layering in details.



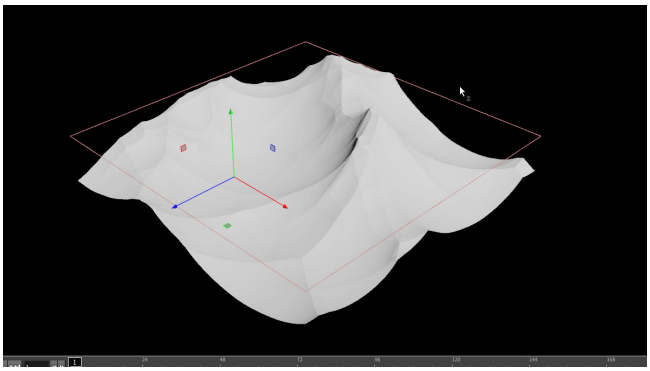
01 Go to the **Desktop** selector and choose **Terrain**. This will give you shelf tools and radial menus focused on terrain. Turn off the **Reference Plane** using the button at the top of the **Display options** then **press d** in the viewport and from the **Background** tab set **Color Scheme** to **Dark**.

Select **File > New Project.** Set the **Project Name** to *terrain_lesson* and press **Accept**. Select **File > Save As...** You should be looking into the new *terrain_lesson* directory. If not then click on $JOB in the sidebar and you will be looking at the directory. Set the file name to *terrain_01.hip* and click **Accept** to save.



02 From the **Terrain Tools** shelf, click on the **HeightField** tool. Press **Enter** to place it at the origin. Press **Spacebar H** to show the whole heightfield.

This defines a 1000 x 1000 grid with a grid spacing of 2.

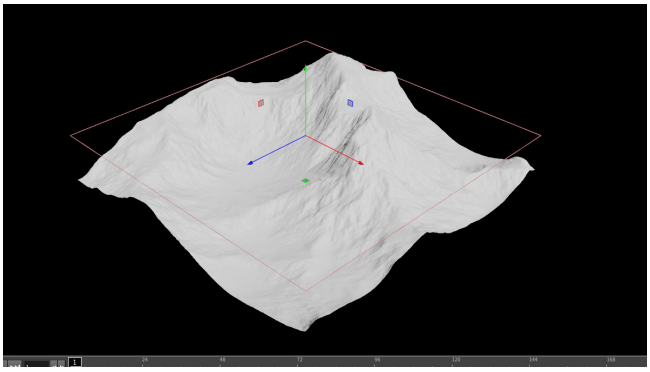


03 Use the main radial menu (**hotkey c**) to select **Deform > Noise**. Set the following:

- **Noise Type** to **Worley Cellular F1**
- **Amplitude** to **360**
- **Offset** to **20, 0, 300**

This kind of noise gives you a good starting point for your terrain.

Use the main radial menu to choose **Deform > Blur** to access the **Heightfield Blur** tool. Set the **Radius** to **20**. This softens the edges to make them feel worn by time.

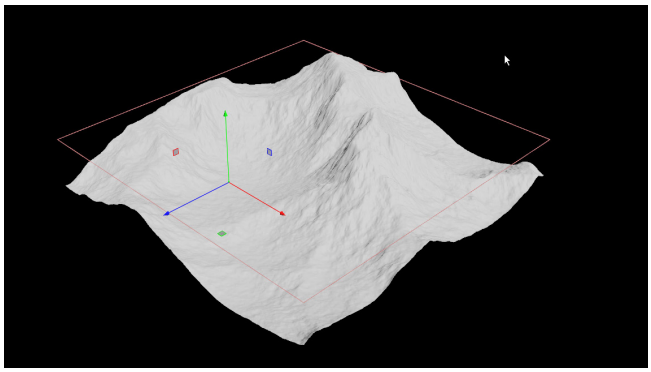


04 Use the radial menu to choose **Deform > Distort**. Set the following:

- **Amplitude** to **40**
- **Element Size** to **220**

This node moves the existing values around by advecting them through a noise field. You can use the parameter values proposed here or explore on your own to get a look that you like better.

Houdini's procedural approach would even allow you to come back later and change the parameter values to see how different settings affect the outcome.



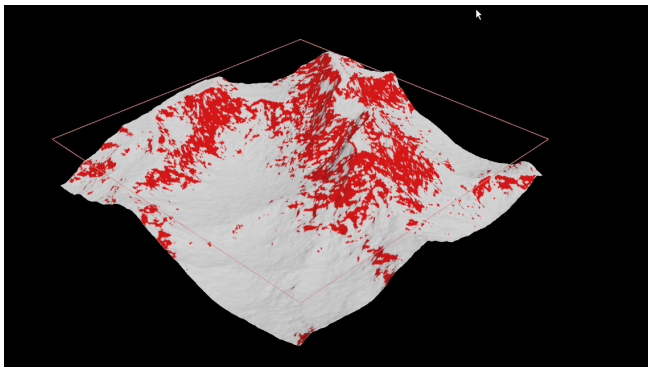
05

Use the radial menu to choose **Deform > Noise**. Set the following:

- **Amplitude** to **10**
- **Element Size** to **20**.

Select the last four nodes in the Network view then click on the **Network box** icon to add a Network box. Adjust its edges to shape the box then click in the top bar and name it *Shape the Terrain*.

Network boxes make it easier to read the network especially if you share your file with other artists.



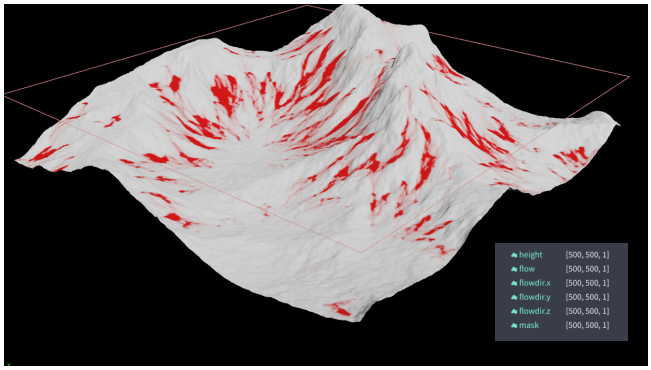
06

Use the radial menu to choose **Mask > Mask by Feature**. Set the following:

- **Min Slope Angle** to **35**
- **Max Slope Angle** to around **60**

This lets you focus on areas on the side of the mountain.

If you were to go back and change the shape of the terrain feeding into this node then the mask would update accordingly.

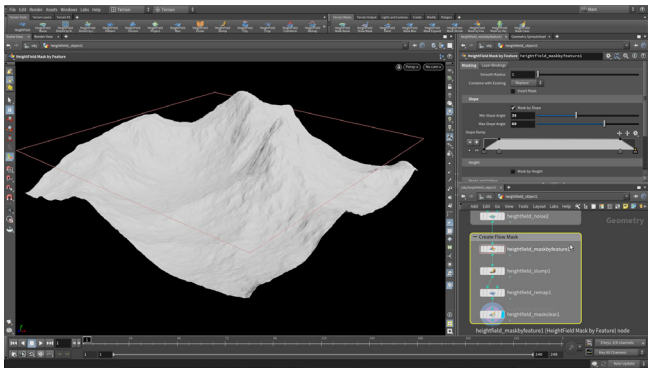


07

From the radial menu, choose the **Erode > Slump**. Set **Spread Iterations** to **75**.

The *heightfield_slump* node creates a type of erosion that moves unstable piles of rubble into a more stable configuration. It affects the Mask layer and also outputs to the **Flow** and **Flow Direction** layers.

If you **MMB-click** on the node, you can see which layers are being created. You can **MMB-click** on an earlier node in the chain for comparison.



08

From the **Terrain Tools** shelf, choose the **Heightfield Remap** tool. Set **Layer to Remap** to **Flow** then click on the **Compute Range** button. Set **Output Max** to **1** to normalize these values. Use the radial menu to choose **Mask > Clear Mask**. This clears the mask layer so that you can use it to create more layers in your setup.

Select the last four nodes in the Network view then click on the **Network box** icon to add a Network box. Adjust its edges to shape the box then click in the top bar and name it *Create Flow Mask*. **Save** your work.

HEIGHT LAYERS

The data passing through a geometry network can contain multiple height fields. In the terminology of Houdini's terrain tools, these are called height layers. For example, a tool might use one height layer to represent bedrock and another to represent loose soil on top. The default height layer is named height.

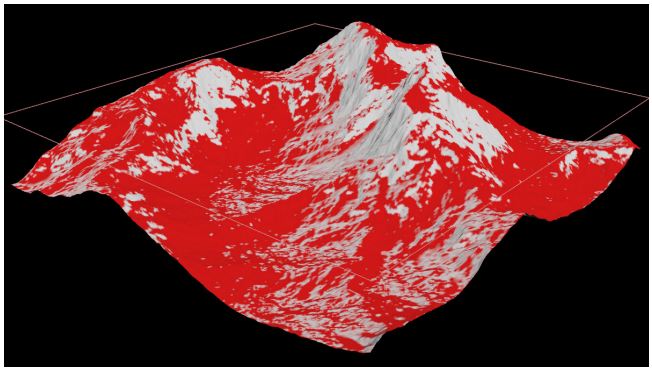
Each voxel contains a "selected-ness" value, called a mask layer. Most terrain nodes take a second input that can contain a mask layer to control which parts of the terrain the node will modify. The default mask layer is named mask and is displayed as a red tint on the 3D surface.

height	[500, 500, 1]
flow	[500, 500, 1]
flowdir.x	[500, 500, 1]
flowdir.y	[500, 500, 1]
flowdir.z	[500, 500, 1]
mask	[500, 500, 1]

PART TWO

Add and Visualize Mask Layers

You can set up layers on your terrain by first populating the mask then copying that information to a particular layer. You can do this more than once to add more layers. These layers can be used later to visualize key aspects of the terrain.



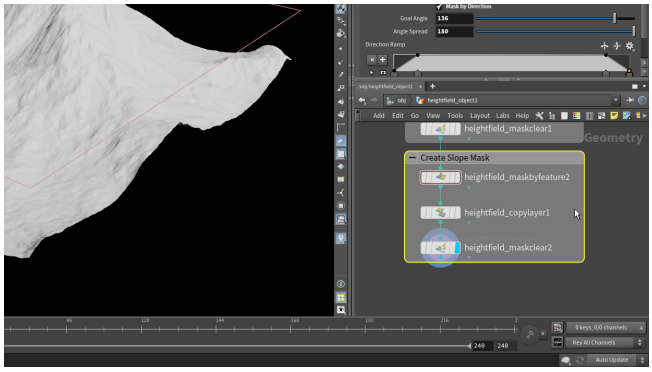
01 From the radial menu, choose **Mask > Mask by Feature**. Under **Mask by Slope,** set the following:

- **Min Slope Angle** to **0**
- **Max Slope Angle** to around **45.**

Turn on **Mask By Direction** then set:

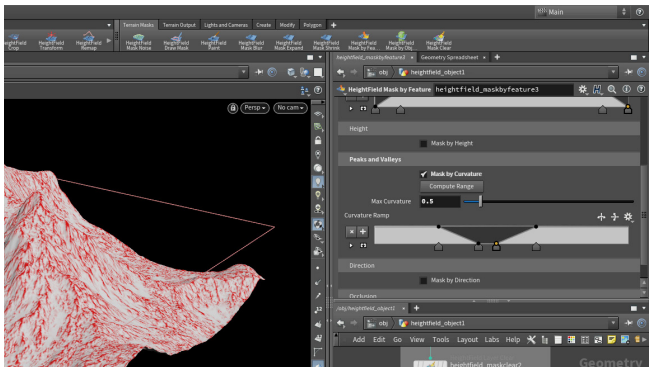
- **Goal Angle** to around **136**
- **Angle Spread** to **180**.

These settings cover a larger area of the terrain including the valleys.



02 From the radial menu, choose **Layer > Copy Layer**. Leave the **Source** set to *Mask* and set the **Destination** to *slope*. By copying the mask to a new layer, it leaves you free to clear the mask and use it for other tasks.

From the radial menu, choose **Mask > Clear Mask**. This again clears the mask layer so that you can use it to create more layers in your setup. Add a network box to organize your nodes and call the box *Create Slope Mask*.



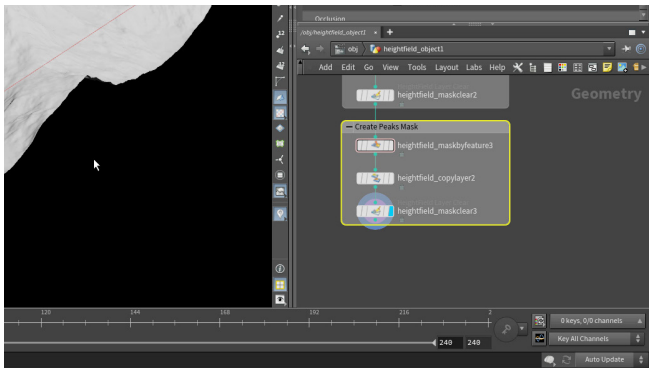
03 Use the main radial menu to select **Mask > Mask by Feature**. Under **Mask by Slope,** set the following:

- **Min Slope Angle** to **0**
- **Max Slope Angle** to around **70.**

Turn on **Mask By Curvature** and set**:**

- **Max Curvature** to **0.5.**

Next move the **Curvature Ramp** points in towards the center to find the peaks of the terrain. This gives you a very detailed mask that you can use to find the peaks of all the key features in the landscape.



04 From the radial menu, choose **Layer > Copy Layer**. Leave the **Source** set to *Mask* and set the **Destination** to *peaks*. Afterwords, choose **Mask > Clear Mask**. This again clears the mask layer.

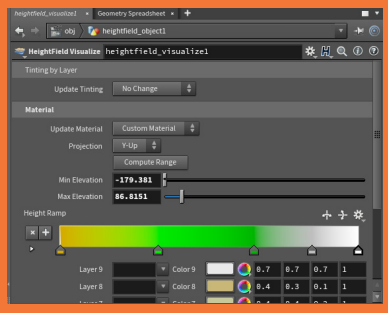
You now have three layers that have all been derived from masks. Add a network box to organize your nodes and call the box *Create Peak Mask*.

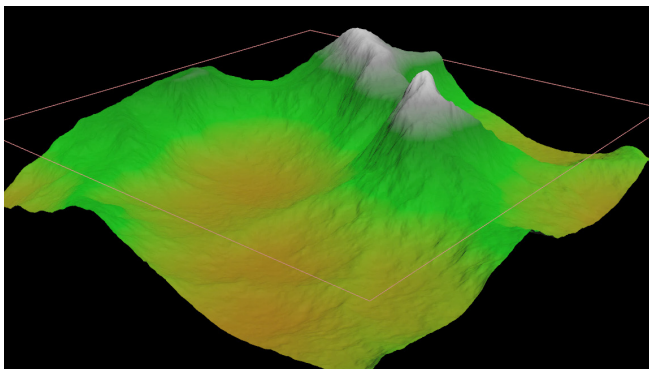
You will use these in the next step to visualize the terrain.

VISUALIZING HEIGHTFIELDS

To visualize your heightfields, you start with a ramp that is assigned to the overall height of the terrain. You start by **Computing the Range** that the ramp will apply to then you can adjust the colors in the ramp itself to visualize your landscape.

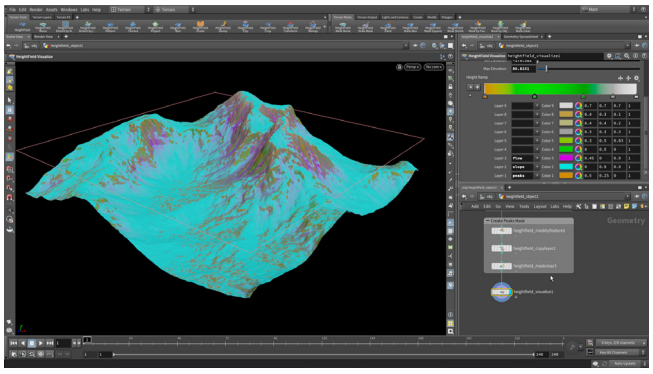
You can then add colors to the various layers that will sit on top of the ramp. This lets you create a richer look for your scene.





05 From the radial menu, choose **Visualize > Heightfield Visualize**. Click on the **Compute Range** button to align the visualization with the current heightfield range.

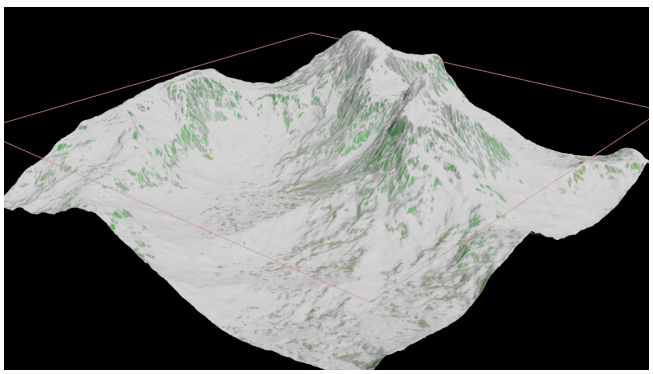
This sets the ramp visualizer to go from the base of the terrain to its peaks. You can see how the ramp looks in the 3D view where the mountain tops are highlighted in white.



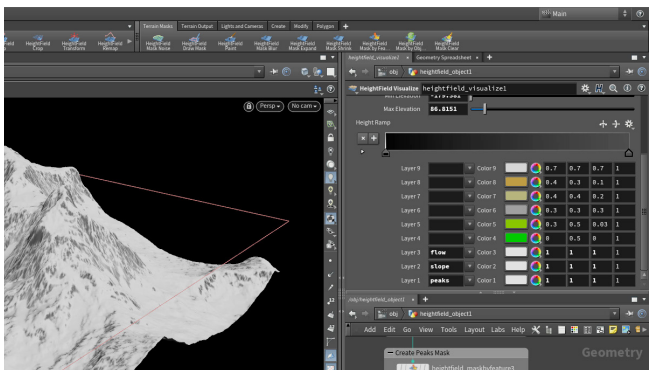
06 Under the ramp widget, set the following:

- **Layer 1** to *peaks*
- **Layer 2** to *slope*
- **Layer 3** to *flow*

The default colors are now visible in the 3D view using the three layers you built up using masks. You will now use these to define the look of the terrain.



07 Set all three of these layers to **white**. [1, 1, 1] This gives a snowy look to the landscape. You can use these layers for any number of different features, but for this mountain, snow is the look that you want to emphasize.



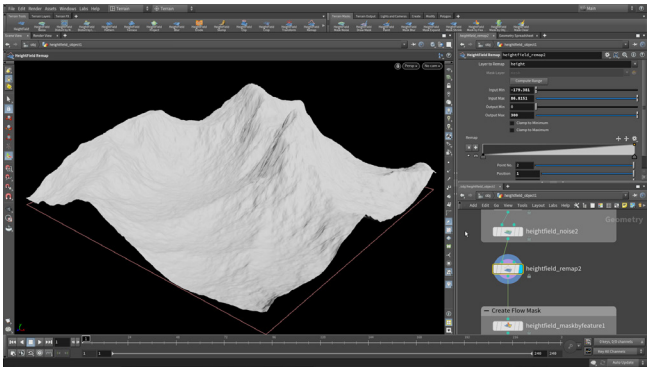
08 In the **Height Ramp**, select and remove all the markers except two. Set the one on the right to black and the one on the left to dark grey. This creates a darker look under the snowy layers which helps the dark areas pop out visually.

Save your work.

PART THREE

Remap and Erode the Terrain

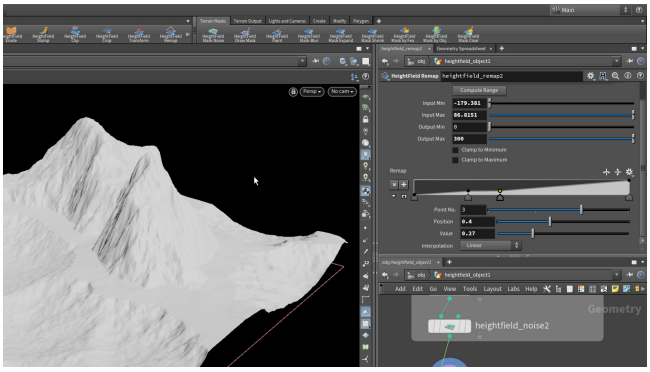
Right now some of the heightmap is below 0 and some is above. You are going to use a Remap node to change the range and then use the ramp on that node to add a ridge around the mountain. You will then erode the landscape to add new layers to the terrain.



01 Find the **Heightfield Noise** node at the end of the Shape by Terrain Network box. Turn on its **Display Flag** and click to select his node.

From the **Terrain Tools** shelf, click on the **Heightfield Remap** tool. Click on the **Compute Range** button. To reframe the heights, set the following:

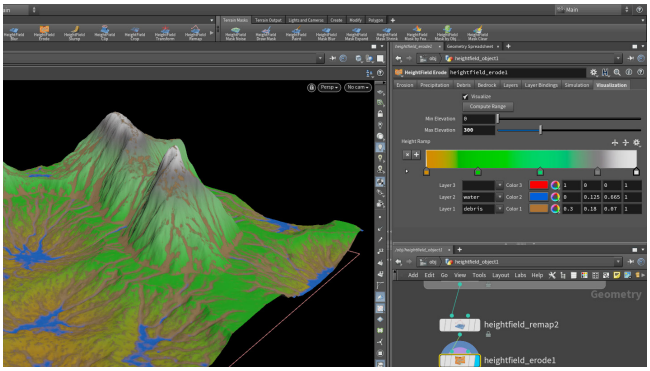
- **Output Min** to **0**
- **Output Max** to **300**



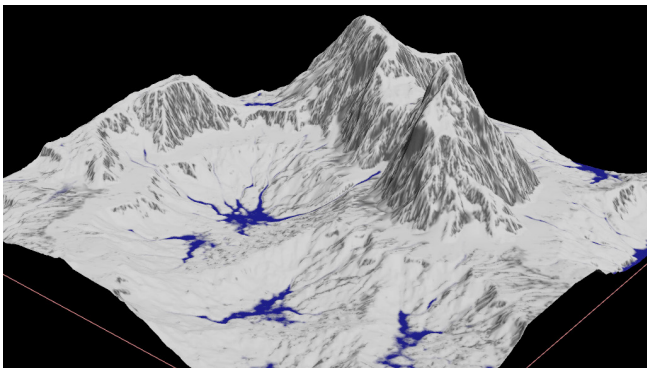
02 Using the **Remap** ramp, add points in the middle then add and adjust the following points:

- **Point 2** has a **Position** of **0.25** and a **Value** of **0.25**
- **Point 3** has a **Position** of **0.4** and a **Value** of **0.27**.

This creates a ridge running along the base of the mountain.



03 Choose **Erode > Erode**. Click on the **Visualization** tab and click the **Compute Range** button. Press **Play** to watch the terrain erode. **Stop** around frame **15**.

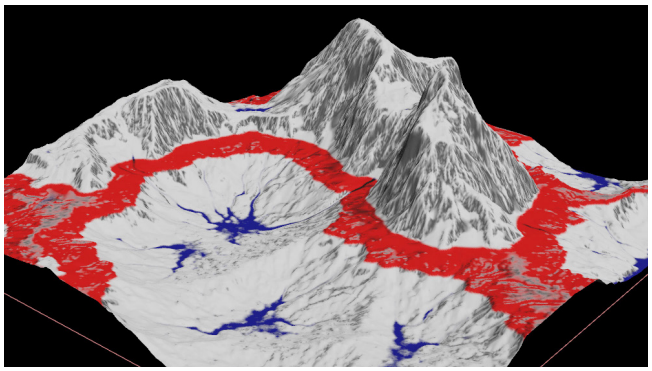


04 Set the **Display flag** back to **Heightfield Visualization** node at the end of the chain. Set **Layer 3** to *water* which is a layer that is coming from the *erode* node and change its **color** to a **blue**. This will bring out some of these areas in the visualization.

PART FOUR

Scatter points on the Terrain

To add trees and rocks, you will mask out the new plateau area then set up a special terrain scatter that will use this mask. These scattered points will then be used to copy instanced cones designed to represent trees. These will be replaced later in Unreal.



01

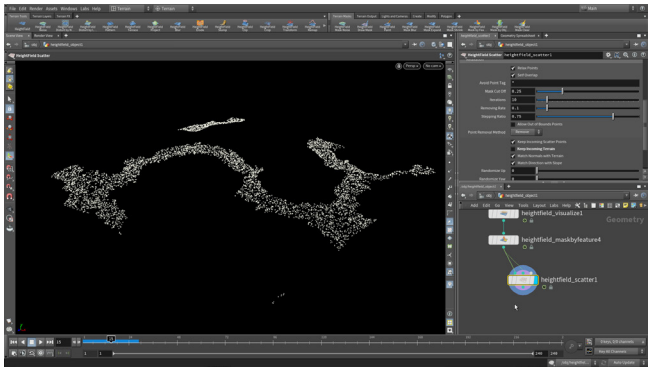
Use the radial menu to choose **Mask > Mask by Feature**. Set the following:

- **Min Slope Angle** to **0**
- **Max Slope Angle** to **50**

Turn on **Mask by Height** and set the following:

- **Min Height** to **70**
- **Max Height** to **85**

This should highlight the plateau created with the *remap* node. If not then tweak these values until you isolate this area in the Mask.

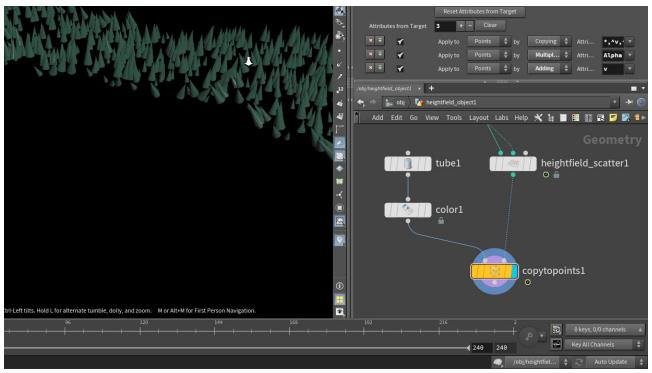


02

In the Network editor **RMB-click** on the output of the *maskbyfeature* node and start typing **scatter...**

Place the **Heightfield Scatter** down and set turn on its **Display flag**. Now wire the output of the *maskbyfeature* node into the second input of the *scatter* node to restrict the points to the area defined by the incoming mask.

Set **Coverage** to **0.05**. Turn **off** the **Keep Incoming Terrain** option.



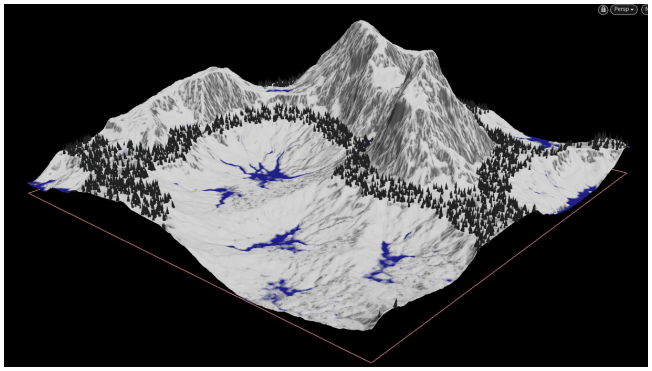
03

In the Network Editor, press **tab** and begin to type **Copy to Points** and click to add this node. Click the **Pack and Instance** check box to turn it on. Wire **Heightfield Scatter** into the second input of **Copy** then set its display node.

In the Network Editor, add a **tube** node down then feed it into the first input of the *copytopoints* node. Set the following:

- **Radius** to **0, 2** and **Radius Scale** to **1**
- **Height** to **10** and **Center** to **0, 5, 0**.

Add a **color** node after the *tube* to make the trees **green**. Add a **Merge** node and feed *Heightfield Visualize* and *copytopoints* into it.



04

Go back to the *heightfield_scatter* node and turn off the **Match Normals with Terrain** and **Match Direction with Slope**. This will ensure that all the trees are pointing up.

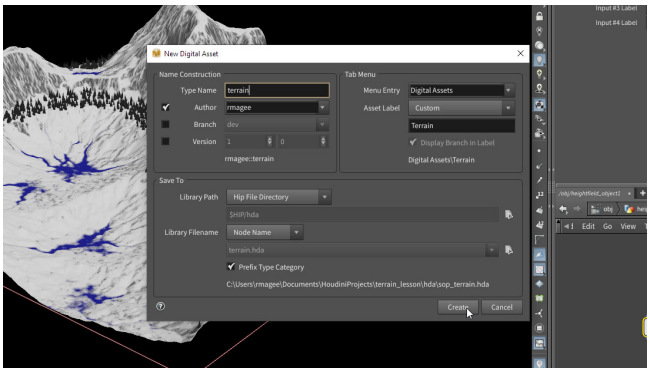
Now change **Random Up** to **10** to create some variation in their direction and set **Randomize Yaw** to **20**. You cannot see the effect of this but if you replace the tree later you will see random rotation.

The Scale of the trees is controlled by **Variability**. Change the **Range** to **1, 2** to create random scales these two values.

PART FIVE

Open the Terrain in Unreal

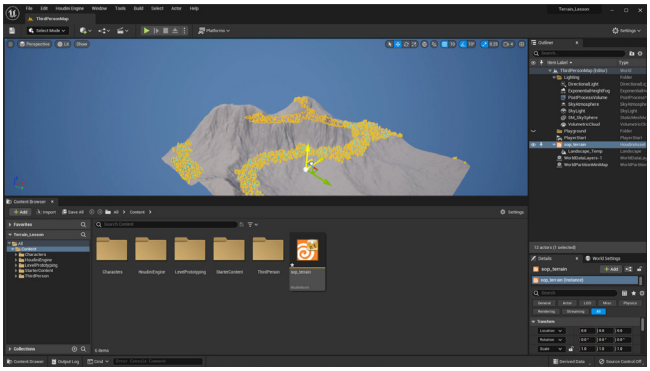
To bring the landscape into game engines such as Unreal Engine or Unity, start by creating a Houdini Digital Asset. Once you have the Houdini Engine plug-in set up properly, then this asset can be loaded into the game editor with the copied tree stand-ins importing as instanced objects. When you import the terrain into Unreal Engine, the heightfields will be recognized as a landscape. You can also import the asset into Unity using the Houdini Engine plug-in.



01 Select all the nodes in the Network editor. Click on the Create **Subnet from Selected Button**. **RMB-click** on the new subnet and choose **Digital Asset> Create New**.

Set the **Type Name** to *terrain* and turn off **Branch** and **Version**. Set **Library Path** to *HIP File Directory* and **Library Filename** to *Node Name*.

Click **Create**. The **Edit Type Properties** window opens up. Click **Accept** to close this window.



02 Open Unreal Engine and from the main panel click on the **New Project** tab and choose the **Third Person** template. Click **Create Project.** When it opens, delete the default geometry so that it doesn't get in the way of your terrain.

From the Content Browser, click **Import to Game** and find the *terrain.hda* asset file. Drag the asset from the **Content Browser** to the 3D workspace. Set the **ThirdPersonCharacter** up to a **Translate Z** of around **10,000** then press **Play** and walk around the terrain.

03 Go to the **Houdini Instanced Inputs** section and expand *terrain1_1*. This is the cone instance which you can replace with content from within Unreal.

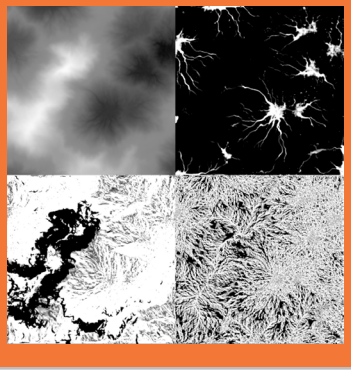
In the Content Browser, open **StarterContent > Props**. Drag the *SM_Bush* prop over to the **Houdini Instanced Input**. Set the **Scale Offset** to **5, 5, 5**. The geometry is instanced to the points then randomly scaled and rotated just like the cones.

In the outliner, select the **Landscape** node under **terrain**. Beside **Landscape** material, click on the menu and find a grass material. Press **Play** to explore the Terrain.

CONCLUSION

You can also use the terrain layers to create texture maps that you can work with to define the look of your landscape. You can do this using the **Heightfield Output** node. You can then use these channels to build a material in the Unreal Engine that references the features of the landscape.

This quick look at the terrain features in Houdini will open up a wealth of possibilities for artists creating landscapes for their games then populating them with details such as rocks and trees.



CHARACTER FOUNDATIONS

KINEFX RIGGING | FUR DUDE

In this lesson, you will rig, animate and add fur to a two legged character named Fur Dude. Starting with existing geometry, you will draw the skeleton, capture the geometry then build rig controls for an animation rig. You will then keyframe a walk cycle and add fur to the surface of the creature.

This lesson uses **KineFX**, Houdini's new SOP-based procedural rigging tools. While these tools are used primarily for retargeting workflows they also include tools for rigging characters and creatures. These tools are still evolving and this lesson offers a taste of what will be currently possible. In future releases, you will see the KineFX and animation workflows expanded and refined.

LESSON GOAL

To rig, animate and add fur to the fur dude creature.

WHAT YOU WILL LEARN

- How to build a skeleton using **KineFX** joints
- How to capture deforming and rigid geometry to the skeleton
- How to wrap up the capture rig into a digital asset
- How to add controls and build an animation rig
- How to animate a walk cycle
- How to add fur to the creature
- How to render using Solaris and Karma.

LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 2.00 | Nov 2022

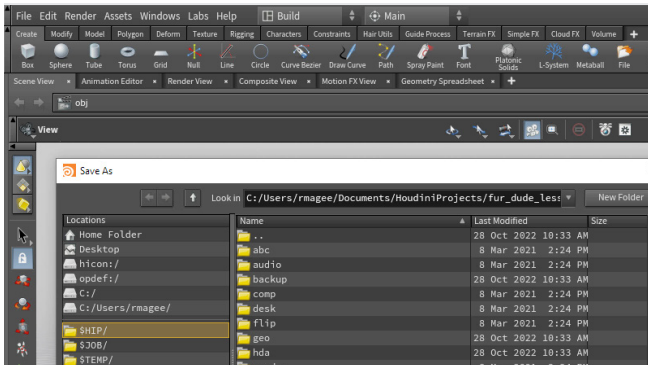
PART ONE

Draw the Skeleton

Start by opening the scene file and reviewing the Fur dude geometry then place joints using the Skeleton tool. This tool will let you create, name and adjust joints to line up with the character you want to animate.

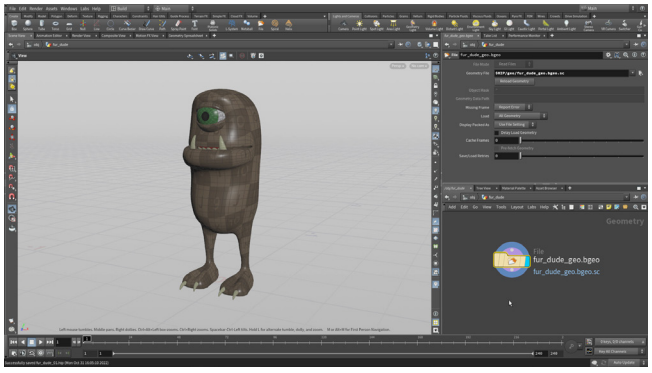
PROJECT FILES

Go to the **Fur Dude tutorial** page on **SideFX.com** where you got this document and download the *furdude_lesson_start* directory. Rename it *furdude_lesson* then put it into the **Houdini Projects** directory.



01 Select **File > Set Project.** Find the *furdude_lesson* directory that you downloaded earlier and press **Accept**. This makes this project directory and its sub folders the place for all the files associated with this shot.

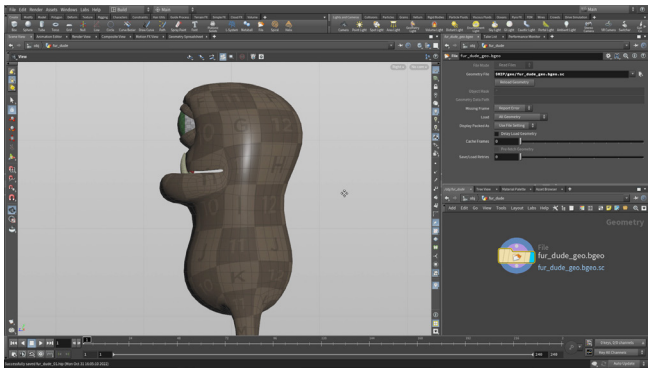
Select **File > Open.** You should be looking into the new *furdude_lesson* directory. Open the file name to *furdude_start.hip*. Select **File Save As...** and rename the file *furdude_01.hip*. Click **Accept** to save. This way you can go back to the start file later if you want to redo the lesson.



02 The scene opens with a single object called *fur_dude_rig*. You are going to capture this geometry to bones created using the **KineFX** toolset.

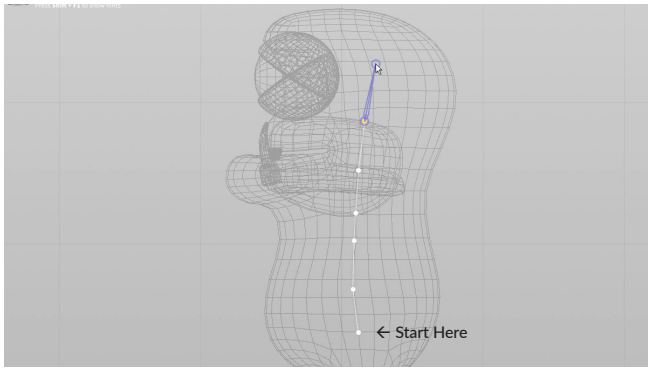
Double-click on the node to dive down to the geometry level. Here you can see the **File** node which is importing the *fur_dude_geo.bgeo* file from disk.

This geometry has some information stored with it such as primitive colors and groups. To see this you can **MMB-click** on the node to see Attributes and Groups listed. You will use the groups later on to help capture the geometry.



03 Move your cursor over the Scene View and press **spacebar-b** to go to a four view layout. From the icon in the top right, turn on **Link Ortho Views**. Now you can pan and zoom in the top, front and right views and they are all synced up.

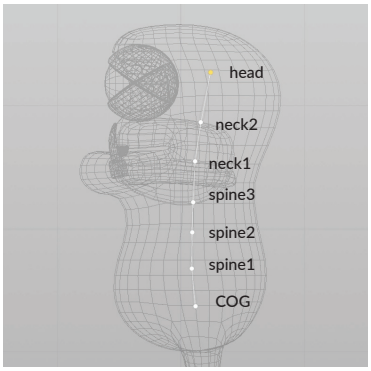
Move your cursor over the *Right* view and press **Spacebar-b** again. This is a good view for drawing joints.

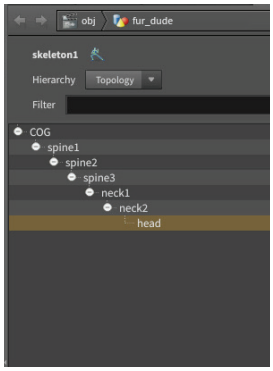


04 In the Network view, press **tab > Skeleton** and place the new node next to the *File* node. Set it's **Display Flag**. Set the file node's **template flag** so that as you are working with the skeleton node the geometry is visible as a grey wireframe.

Make sure the Handle tool is selected. In the top bar, set **Joint Placement** to **Freehand**. This will draw on the construction plane without taking the geometry into account. Click to place your first joint just above the leg and then place six more joints as shown in this picture.

MMB-click to stop drawing joints.

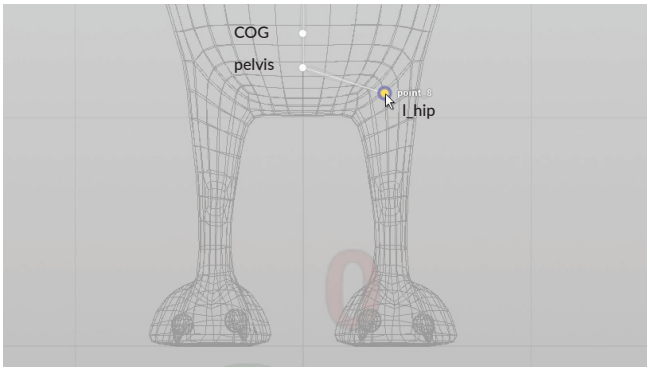




05

In the top bar, set **Mode** to **Modify**. Now you can edit the joints instead of drawing them. Click on the first joint and in the top bar set its **Name** it *COG*.

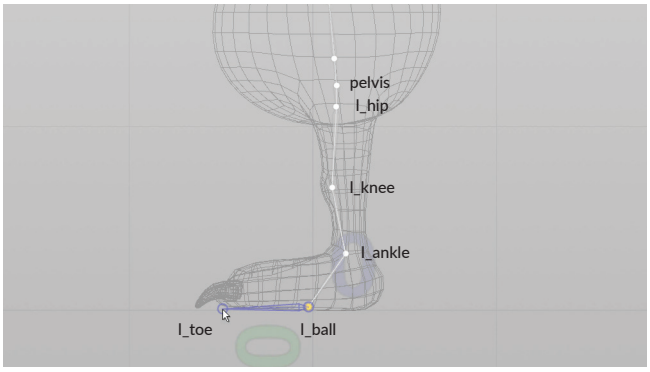
In the Parameter pane, click on the **+ sign** in the tab area. From this choose **New Pane Tab Type > Animation > Rig Tree**. This brings up a pane that shows your skeleton joints. You can double click on the second joint and name it *spine1*. Now either use the Scene View or the **Rig Tree** to name the rest of the joints as shown.



06

In the top bar, set **Mode** back to **Create**. By default it will try to draw from the end of any selected joint. **MMB-click** to stop this drawing action. Now click on the *COG* joint in the Right view then draw a *pelvis* joint just below it.

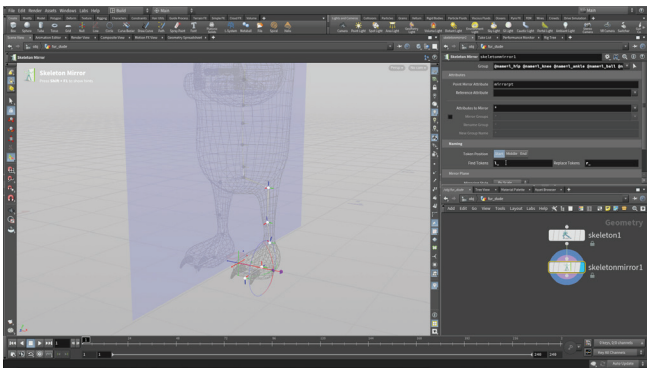
In the Scene View, press **spacebar-b** to go back to a four view In the Front view, draw a *hip* joint on the left side of the character.



07

Now go back to the Right view and draw the final four joints for the leg as shown here.

Go back to **Modify** mode then rename the joints using either the **Rig Tree** or by selecting and naming the joints in the top bar. After the *pelvis*, all the other joints will have a "*l_*" prefix since these joints will be used for the left leg.



08

In the Scene View, press **tab > Skeleton Mirror**. This creates a mirror copy of all the joints. Go to the Parameter pane and click on the arrow next to **Group**. Select only the leg joints and press enter. Now only the leg is being mirrored.

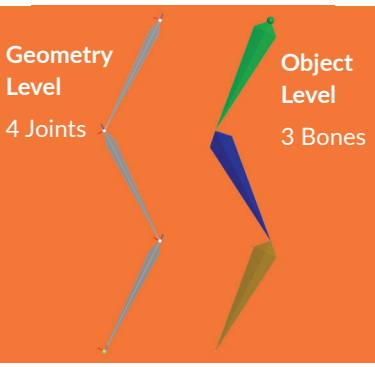
Under naming, set **Find Tokens** to *l_* and **Replace Tokens** with *r_*. Now you have the right leg joint properly named.

Select **File > Save** to save your work so far.

KINEFX VS OBJECT LEVEL RIGGING

The KineFX tools in Houdini offer a joint-based workflow that takes place at the geometry [SOP] level. Houdini's other character workflow is bone-based and in this case you work primarily at the object [OBJ] level.

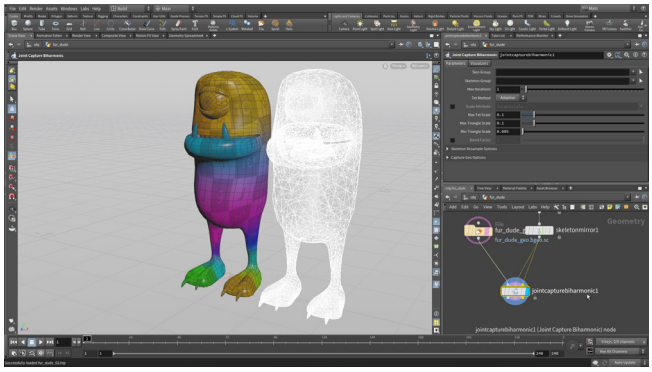
In the KineFX workflow, joints are basically just points on a curve and this opens up lots of opportunities to use sop-level tools to manipulate rigs. Here you are going to learn about tools designed specifically for rigging characters and creatures.



PART TWO

Capture the Geometry

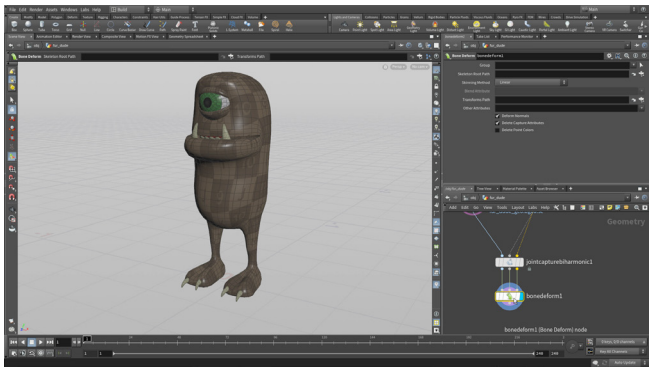
Rigging a character involves capturing geometry to the skeleton joints in such a way that rotating the joints deforms and bends the geometry. Houdini uses a biharmonic capture method that gives great results with your first capture so that you can start testing out your rig right away. Later you will paint capture weight s to refine the results to work with your character.



01 Change your Scene view to the perspective using **spacebar-b**. Make your Network View a bit bigger so you have room to work.

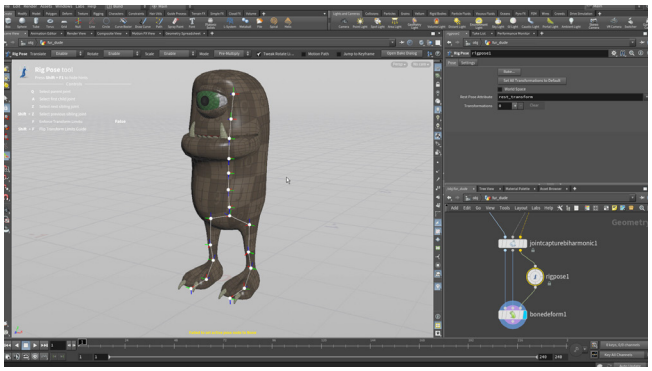
In the Network view, press **tab > Joint Capture Biharmonic** and place this node down under the skeleton nodes. Wire *fur_dude_geo* into the *jointcapturebiharmonic* node's **first input**. Wire *skeletonmirror* into the *jointcapturebiharmonic* node's **second and third input** and then set its **Display flag**.

You can now see capture weights on the geometry. You will refine and paint these later to set up the deformation of the geometry.

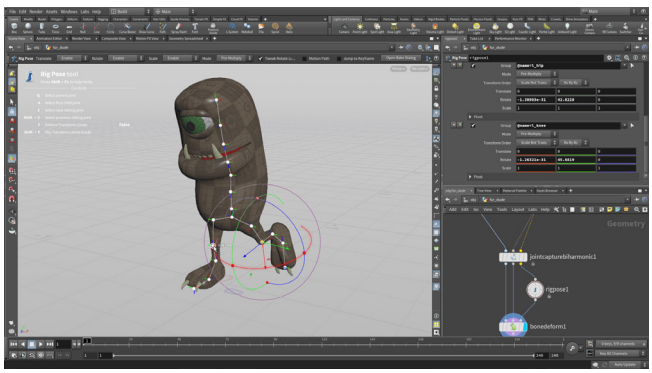


02 In the Network view, press **tab > Bone Deform**. Wire the **three outputs** of *jointcapturebiharmonic* into the **three inputs**. Set the **Display flag** on *bonedeform*.

In this lesson, you are capturing teeth, claws and the eye which you probably don't want to deform. You will split these out later to capture them using a different method.



03 Next press **tab > Rig Pose** and place down the node. Move it over the third line connecting *jointcapturebiharmonic* and the third input on *bonedeform* to add it into the chain. This is where animation goes on the rig and the rigpose will be used to rotate joints and can also be used to set keyframes.

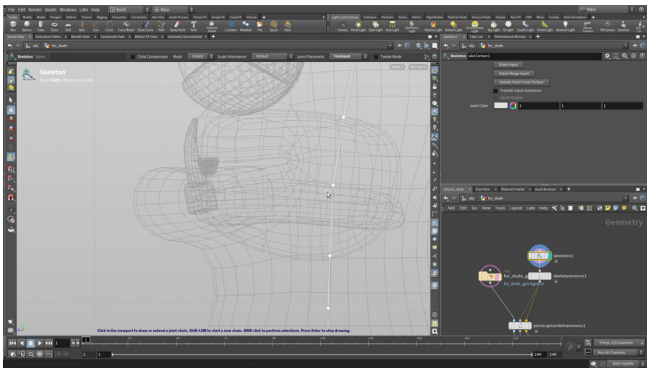


04 Select the *rigpose* node and make sure the **Handle** tool is active in the Scene View. **Select** and **rotate** various joints to test out the deformation. You can reset this later so feel free to explore.

PART THREE

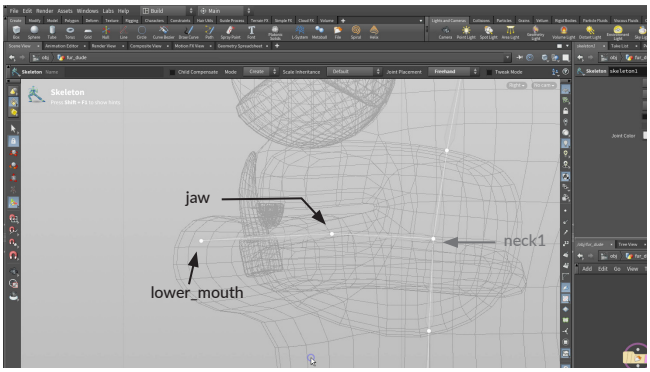
Add More Bones

It would be nice to have more bones in the mouth area. The procedural networks in Houdini let you to go back and add the joints and all the other nodes including the biharmonic capture will update to reflect the changes. This gives you flexibility when first setting up your creature's rig.

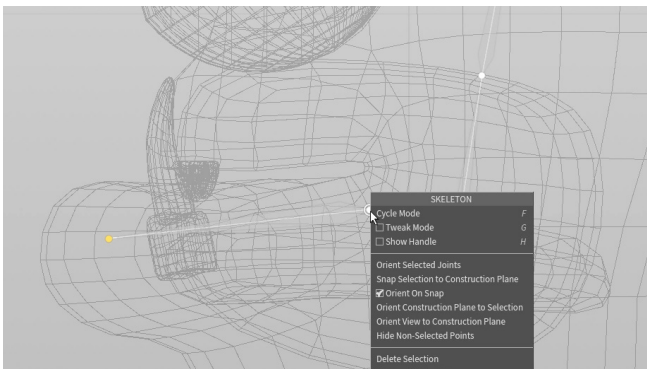


01 In the Network view, set the **Display flag** on the skeleton node and the **Template flag** on the original *File* node. Go to the Right view in the Scene View using **spacebar-b** twice.

Set the **Display Flag** on the *Skeleton* node. Select the *skeleton* node and turn on the **Handle** tool. Set **Mode** to **Create**. This will allow us to add more joints to the skeleton. With KineFX, you can add bones and the procedural nature of the other nodes in our network will allow these changes to be accepted.



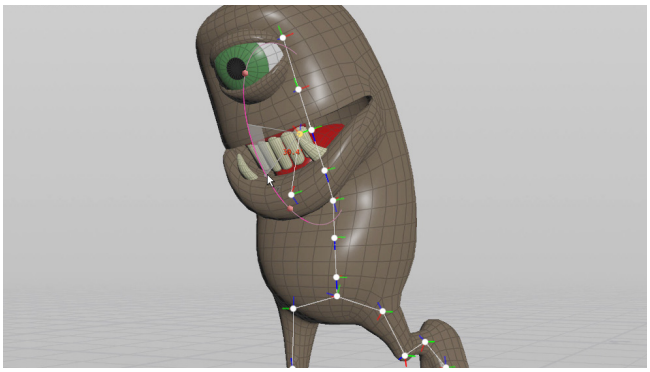
02 Click on the *neck1* joint to start drawing then click two joints at the jaw and lower mouth. When you have these joints in place **MMB-click** to stop drawing and change **Mode** back to **Modify**. Now either click on the joints and rename them *jaw* and *lower_mouth* or rename them using the **Rig Tree** view.



03 You can edit the joint positions in **Modify** mode. Turn **On** the **Tweak** checkbox in the top bar and now you can **click drag** on joints to move them. As you move a joint, all of its children will also move and you may have to adjust them back into place. To avoid this, you can turn on **Child Compensate**.

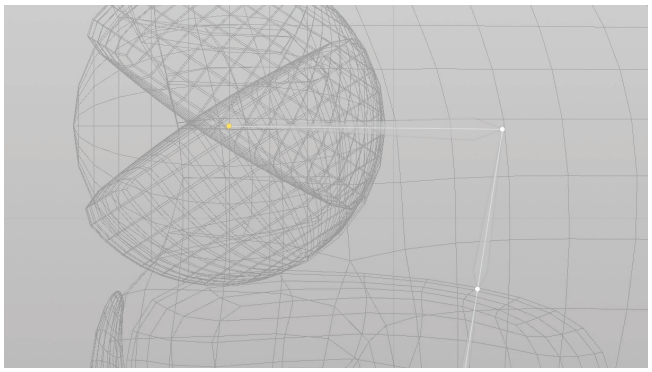
If you **RMB-click** on a joint, you will see options for splitting, unparenting, copying and pasting joints. You won't need to do those things for this skeleton but it is good to know they are there.

You can even mirror joints here but you are using a different node for that in this network.



04 Set the **Display flag** on the *bonedeform* node and turn off the **Template flag** on the *file* node. The geometry will be reconfigured then captured using the new bones. Select the *rigpose* node and select the **Handle** tool.

Click on the new *jaw* joint and rotate it down. This works but it manipulates both the lower lip and upper lip. It would be better if it only affected the lower lip. To fix this you will paint the capture weights in a later section of the tutorial.

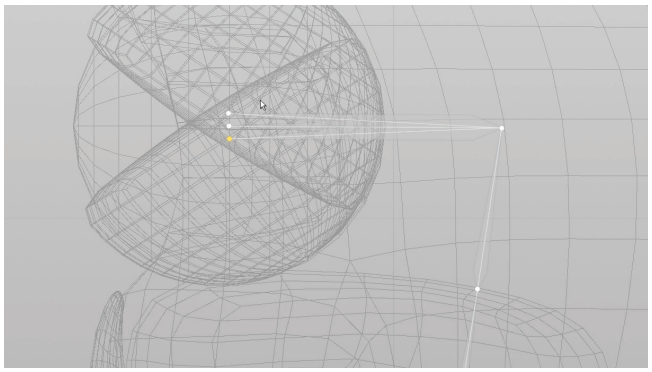


05

Set the **Display flag back to** the skeleton node and the **Template flag** on the original *File* node.

Select the *skeleton* node and turn on the **Handle** tool. Set **Mode** to **Modify**. Click on the **head** joint and using **Tweak Mode,** lower it until it aligns with the eyeball.

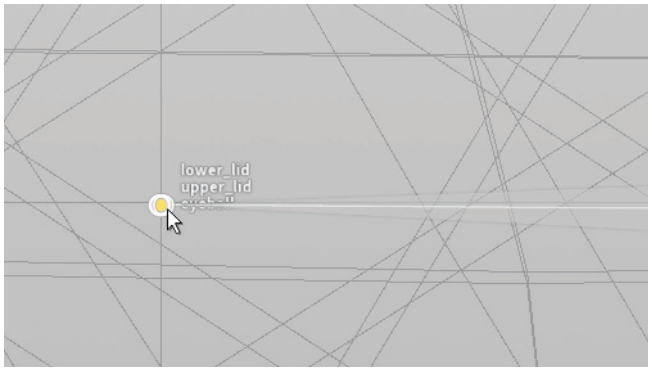
Set **Mode** to **Create**. Click on the *head* joint and then click a new joint right in the center of the eyeball. Switch to **Modify** mode and **Name** this joint *eyeball*.



06

Set **Mode** to **Create**. Click on the *head* joint and then click a new joint above the eyeball joint. **MMB-click** to deselect then again click on the *head* joint and then click a new joint below the *eyeball* joint.

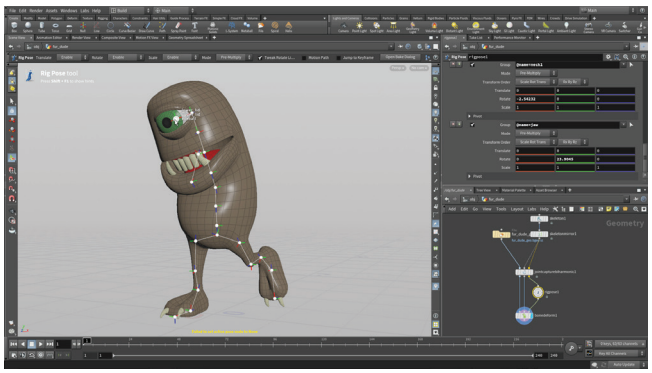
Switch to **Modify** mode and **Name** these joints *upper_lid* and *lower_lid*. You will use these joints to rotate the eyelids during animation but they need to also be at the center of the eye.



07

Turn off the Template flag on the geometry. Select the *upper_lid* joint and use **Tweak Mode** to drag it down on top of the *eyeball* joint. Repeat with the *lower_lid* to overlap the *eyeball* joint.

You now have all three joints in the same spot. Later you will attach geometry to each of them individually and then you will be able to animate them independently.



08

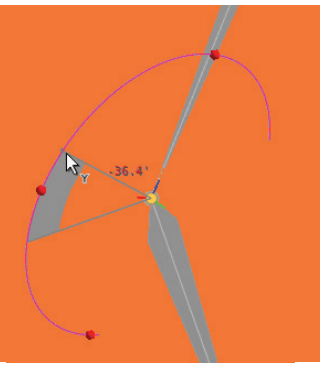
Set the **Display flag** on the *bonedeform* node. The geometry will be reconfigured then captured using all the new bones.

You are not able to pose the new joints yet because the geometry hasn't be attached to them in a rigid manner yet. That will come a couple steps down the line after you paint capture weights on the body and the tongue.

JOINT ORIENTATION

With a joint based system where each joint is a point, the joint orientation is critical because it defines how the joint will rotate during forward kinematics.

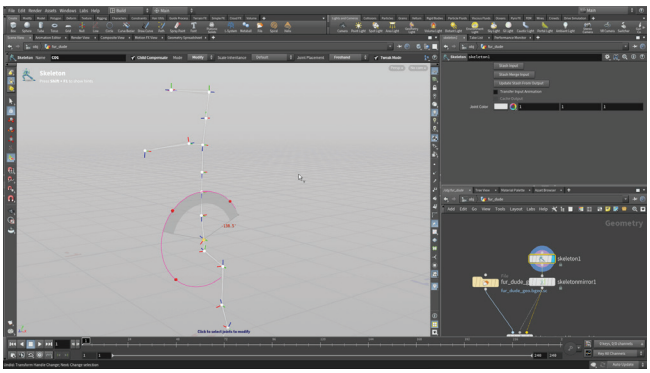
KineFX has tools that allow you to point along the chain to the next joint or you can rotate each joint by hand using the **child compensate** option to make sure that re-orienting the joints doesn't affect the other joints in the chain.



PART FOUR

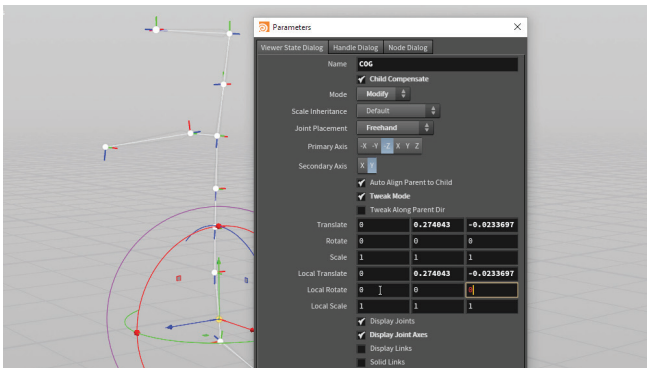
Joint Orientations

When you animate the character, the orientation of the joints plays a significant role in how you manipulate the rig. At this point, you will orient some of the joints by hand and then use a Re-orient joints node to point all the other joints down the -z axis. Sometimes when you evaluate your rig at a later stage, you may need to come back and tweak the orientations.



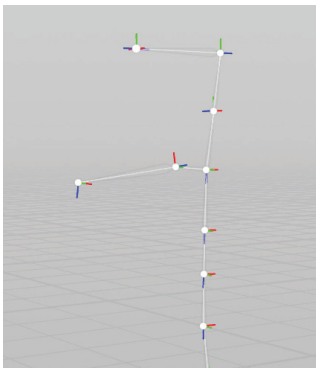
01 Go back and set the **Display flag** on the *skeleton* node. In the Scene View , **RMB-click** and select **Display Joint Axes** to see the joint orientations. Select the *COG* joint and **RMB-click** and select **Show Handle**. Now if you rotate the joint the whole rig goes with it. Press **Ctrl-Z** to undo.

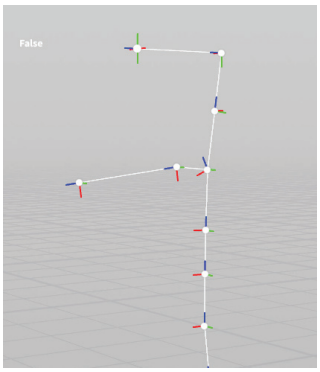
Click on the **Child Compensate** checkbox and now you can rotate the joint without changing the rest of the skeleton. You can use the **Ctrl** key to constrain to 45 degree increments.



02 In the Scene View, **press p** to bring up the parameters for this joint. To orient the COG with the world, Set **Rotate** and **Local Rotate** to **0, 0, 0**.

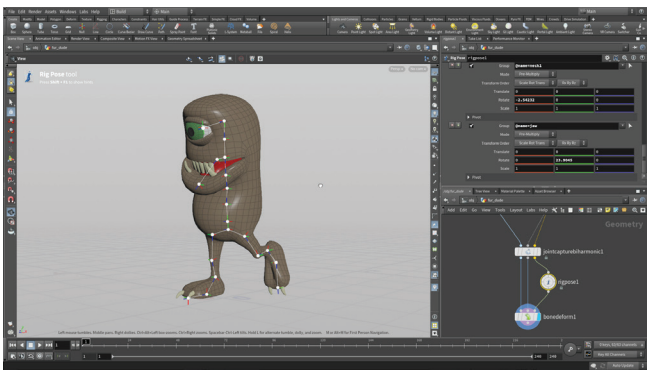
Click in empty space to deselect the COG joint then click on the *Pelvis* joint. Set **Rotate** and **Local Rotate** to **0, 0, 0**.





03 After the Skeleton joint, insert a **Orient Joints** node which will by default point the orientations along the positive Z axis. Click on the arrow next to **Orient group** and in the Scene view, select all the joints. **Press Ctrl** and select the *COG*, *pelvis* and *neck1* joints to remove them from the selection. Press **Enter**.

Now all of the joints are oriented along positive Z except for the three deselected joints.



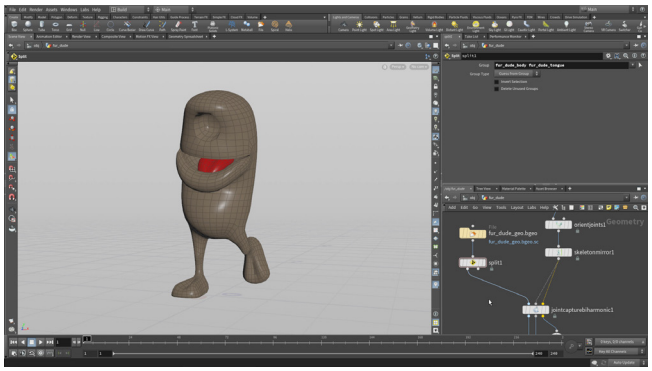
04 Set the display flag back to the **Bone Deform** node to allow all the other nodes to update and accept the new joint orientations.

The results don't look any different than before you oriented the joints but it will affect how you pose and manipulate the character when you are animating.

PART FIVE

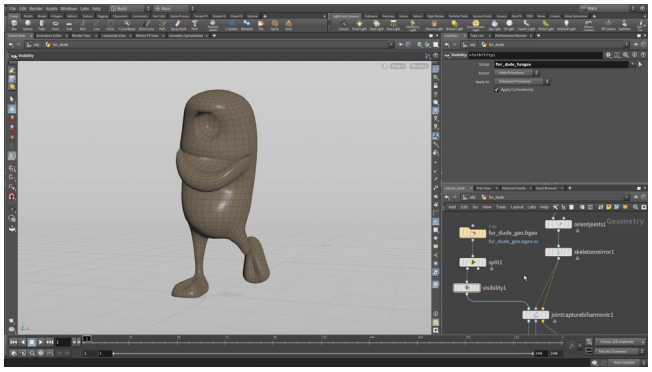
Attach Capture Geometry

Right now you are using the skeleton joints to capture geometry and assign capture weights to each point on your character. To manage this with more control, you can attach curves to the joints that will extend the influence of that joint. This provides a method for achieving your goals as quickly as possible.



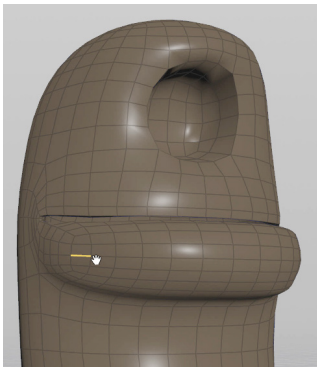
01 In the Network View, press **tab > Split** and place the node between the *File* node and the *jointcapturebiharmonic* node. The first output of the *split* node should be wired into the first input of the *jointcapturebiharmonic* node.

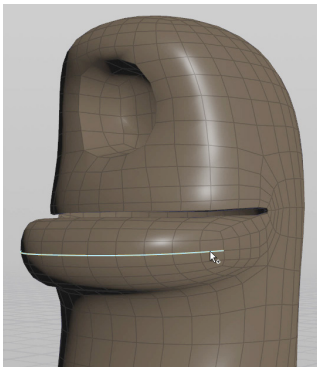
Click on the pull down menu in the **Group** field and choose the *fur_dude_body* and *fur_dude_tongue* groups. These are now feeding the first output of the split node and all the remaining pieces such as the eyeball, teeth and claws are going out the second output. You will use a different method to bind those to the skeleton after painting weights on this geometry.



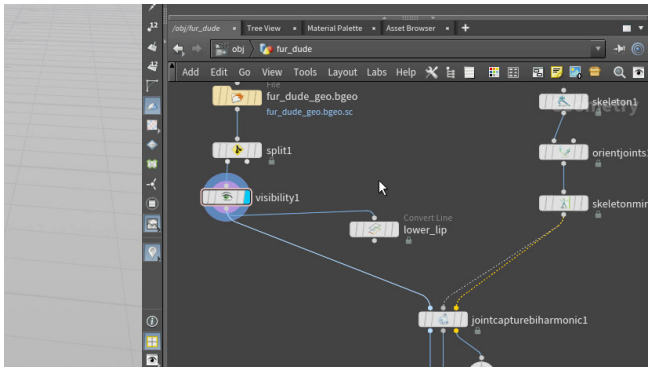
02 In the Network View, press **tab > Visibility** and place the node between the *jointcapturebiharmonic* node and the *bonedeform* node. Click on the pull down menu in the **Group** field and choose the *fur_dude_tongue* group.

This node hides the selected geometry but does not remove it. This ensures that point numbers and primitive numbers don't get changed which is important when working with the paint capture weight tool. You don't want to change this information every time you hide some geometry during this process.



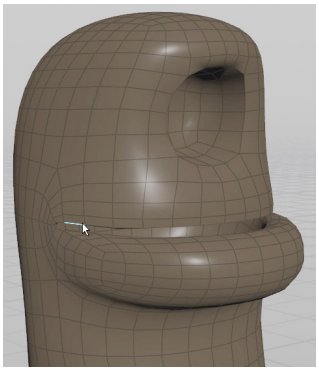


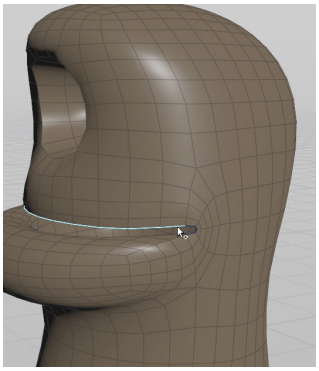
03 **Press S** to get the **Select** tool and **3** to go to Edge selection. Select an edge in the middle of the lower lip on the left then press **Shift-A** and select an edge on the other side.



04 Press **tab > Curve from Edges** and the selected edge will be extracted from the geometry. Rename this node *lower_lip*.

The *curvefromedges* node gets placed in between the *file* and *jointcapturebiharmonic* nodes. Move it to the side and then reconnect the *file* node output to the first input of the *jointcapturebiharmonic* node. This branches the *curvefromedges* node off to the side.

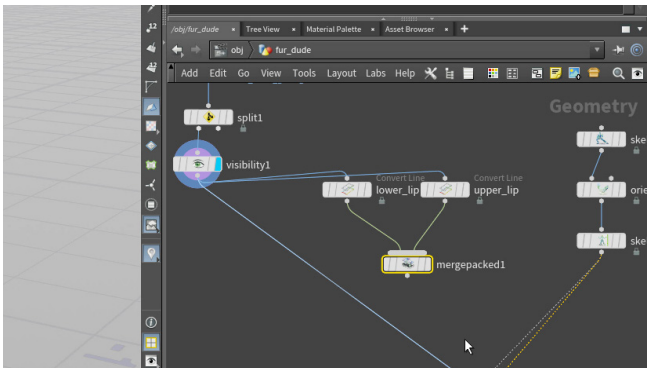




05 **Press S** to get the **Select** tool and **3** to go to Edge selection. Select an edge on the upper lip on the left then press **Shift-A** and select an edge on the other side.

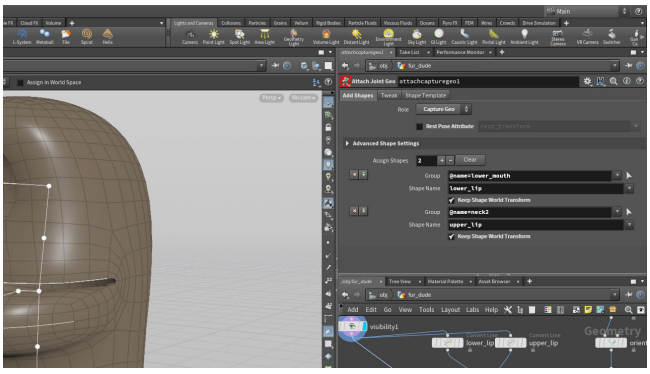
Press **tab > Curve from Edges** and the selected edge will be extracted from the geometry. Rename this node *upper_lip*.

Branch this node off the same as you did with the other one.



06 Press **tab > Merge Packed** and place this node down underneath the two extracted curve nodes then feed the output of those nodes into the *mergepacked* node.

Press **tab > Attach Capture Geo**. Add this node to the network then feed the *skeletonmirror* node into the first input and the *mergepacked* into the third input.



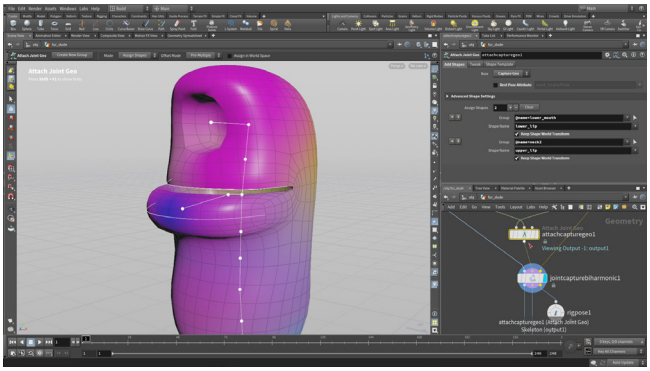
07 Feed the *add_capture_geometry* node's output to the middle input of *jointcapturebiharmonic*. Under **Advanced Shape Settings**, click the **Assign Shapes** plus sign button two times. For the first one, set:

- **Group** to *@name=lower_mouth*
- **Shape Name** to *lower_lip*

And the second one to;

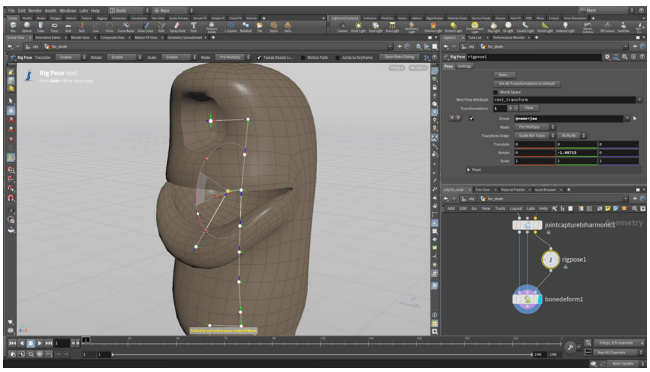
- **Group** to *@name=neck2*
- **Shape Name** to *upper_lip*

Turn **On** keep **Shape World Transform** for both Shapes.



08 Set the **Display Flag** on *jointcapturebiharmonic*. Now **bypass** the *add_capture_geometry* node to show difference in the capture weights.

When this node is turned on, the attached curves are assisting with the capture of geometry to the associated joint. This allows for more control as you set up your character.



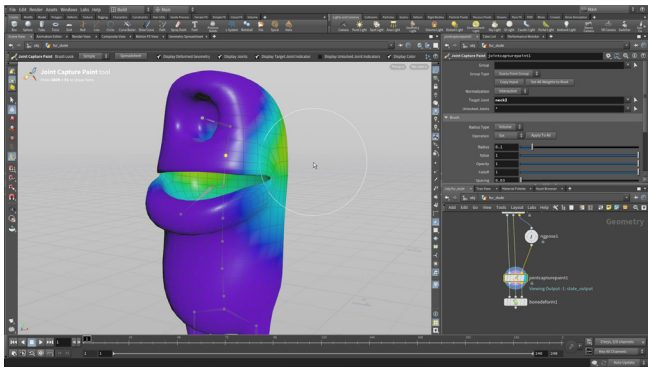
09 Set the **Display flag** on the *bonedeform* node. Click on the *rigpose* node and in the Parameter pane, click on the **Clear** button next to **Transformations**. This will reset the joints. Now with the **Handle** tool active and in the Scene View click on the *jaw* joint. **Rotate** it to lower the lips down.

You can see that this still deforms parts of the belly. To correct this, you need to paint capture weights to reassign capture weights to different joints.

PART SIX

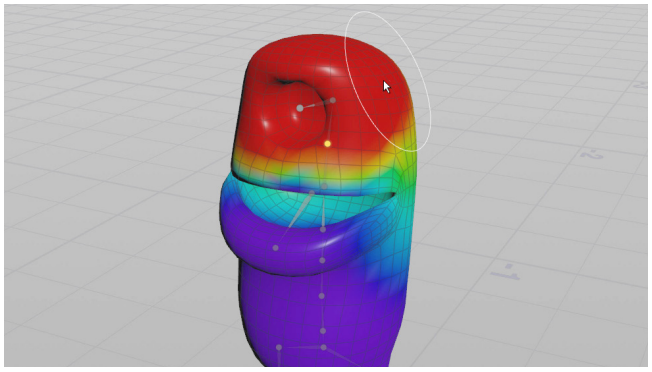
Paint Capture Weights

The biharmonic capture added weights to the character's geometry that are associated with the different skeleton bones. You can now use a new node to adjust the capture weights using a brush workflow. For this creature, the goal will be to get the top of the mouth to not be influenced by the lower mouth joint and to tweak how the feet area is weighted.



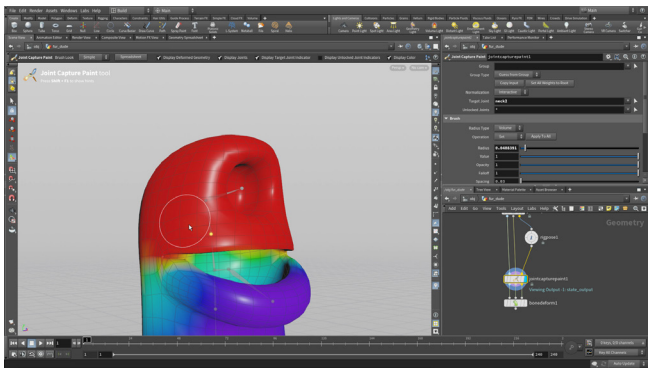
01 In the Network View, press **tab > Joint Capture Paint** and place the node just above the *bonedeform* node with all three connectors getting connected. Set its **Display Flag** then click on the pull down menu next to **Target Joint** and choose the *neck2* joint.

In the Scene view, you will also see a big round paint icon on your cursor which you can use to paint weights. Paint on the head area and this part of the geometry will be captured to the *neck2* joint.



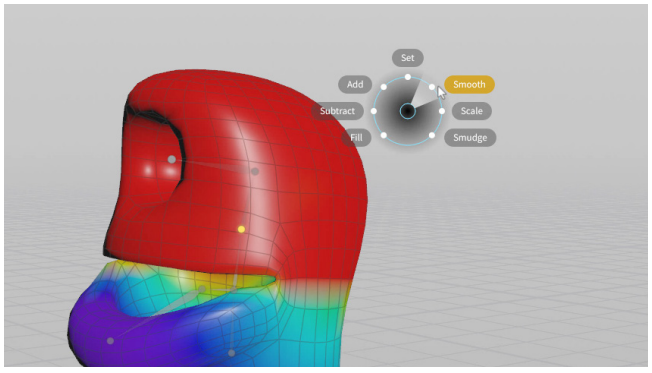
02 You want to capture the head and the upper lip area to the neck2 joint. The geometry will appear red when it is at its strongest.

Use big strokes for the top of the head then use the **Scroll wheel** on your mouse to reduce the brush radius or go to the **Brush** tab in the Parameter pane to change the radius.



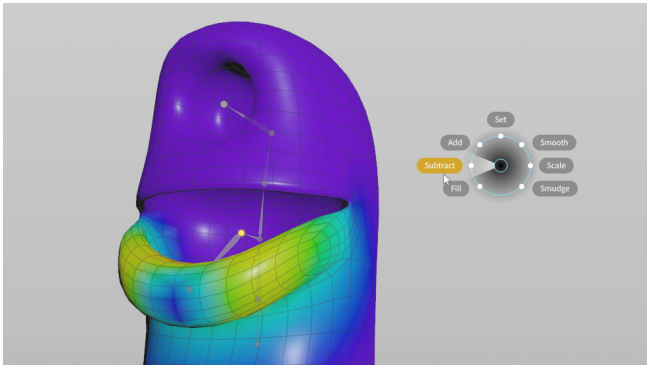
03 Paint on the upper lip to increase the influence of the *neck2* joint on this area. Make sure you are painting the top lip and not the bottom lip. If you mess up, use **Ctrl-Z** to undo your stroke. Tumble around, even inside the mouth to get the upper part of the mouth.

In the **Operation Controls** panel, you can see options to **Display Deformed Geometry**, **Display Joints** and **Display Color**. You can turn these on and off to help you evaluate your capture weights as you paint.



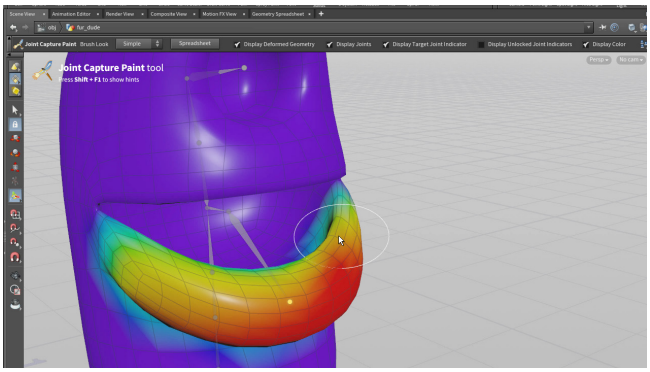
04 **Press F** and choose **Smooth** to smooth out what you have painted here with other capture weights. Make your brush radius a bit bigger to create more smoothing.

Once this is finished, click on the *rigpose* node with the **Handle** tool active and in the Scene View click on the *jaw* joint. Rotate it to lower the lips down. You can see that now the top lip isn't moving but other parts of the head are being rotated.



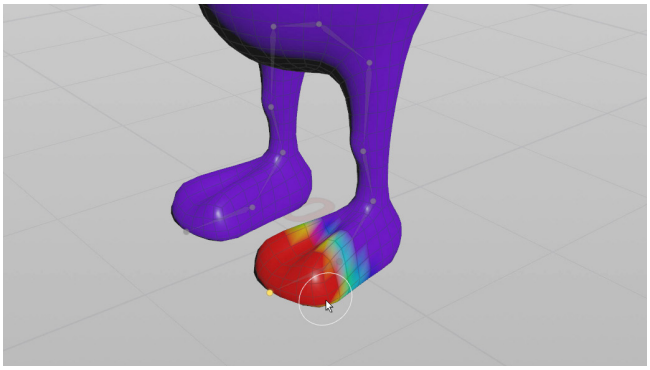
05 Select the *jointcapturepaint* node. Click on the pull down menu in the **Capture Region** field and choose the *jaw* joint. **Press F** and choose **Subtract** to take away influence of this joint on the top of the head and eye area.

You can also use this method to remove the influence of the jaw and the lower_mouth joints on the belly. That way when the mouth moves the belly isn't affected too much.

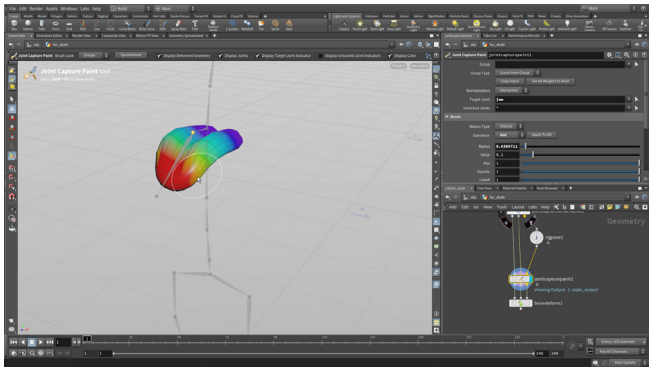


06 From time to time, go back to the and test out the jaw. Once you have it no longer influencing the head area then you are good to go. You can also go back to smoothing to clean things up once you are finished.

You can now paint weights for other joints in the skeleton. The Biharmonic should have done a good job on the legs and feet but you can test the rig using *rigpose* then paint weights to refine the results.



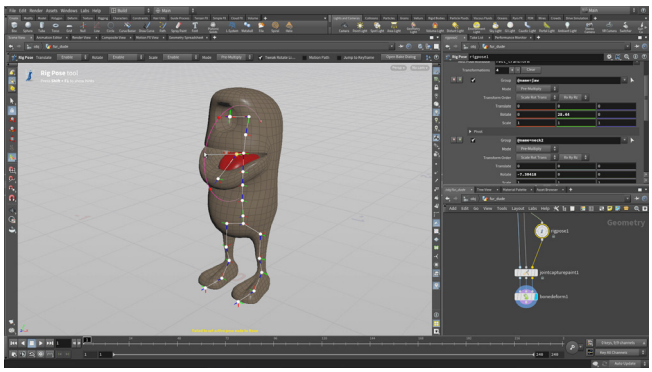
07 Select the toe joints and make sure that the ends of the feet are captured to those joints. Later you will use a reverse setup and having the geometry attached to the toes .



08 On the visibility node set **Apply to Non-Selected Primitives** . Now you only see the tongue.

It should also be weighted fine by default but you can paint weights if you need to. Select the *jointcapturepaint* node and paint weights for the *neck1* and *jaw* joints.

When you are finished then **Bypass** the *visibility* node again.



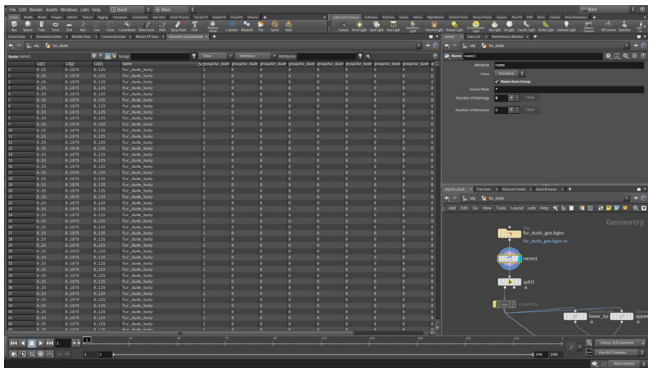
09 On the visibility node set **Bypass Flag**.

Once this is finished, click on the *rigpose* node with the **Handle** tool active and in the Scene View click on the *jaw* joint. Rotate it to lower the lips down. If you are happy with how all the parts are working then you are ready to capture the rigid geometry.

PART SEVEN

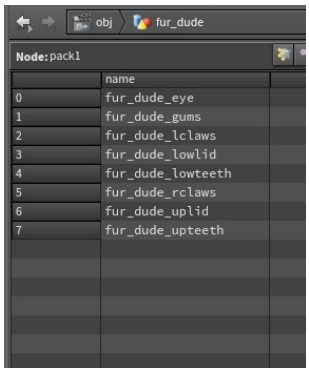
Capturing the Rigid Geometry

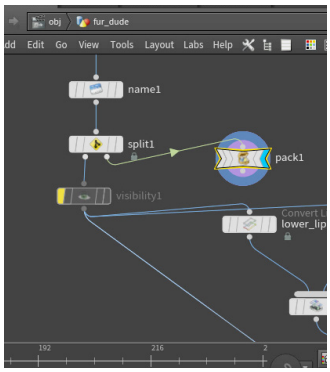
Earlier you split out the eyes, teeth and claw geometry. Now you are going to pack this geometry then assign each part to a joint using the capture packed geometry node. This is the equivalent of parenting each object to the skeleton since parenting isn't an option at the geometry level when using KineFX.



01 In the Network View, press **tab > Name from Groups** and place the node between the *File* node and the *split* node. Change the **Group Mask** to *. This will turn all of the groups into name attributes.

To see this, click on the **Geometry Spreadsheet** Tab and click on the **Primitives** button at the top left. Scroll down to see that there is a name attribute and the group names are used as the values for all of the geometry.

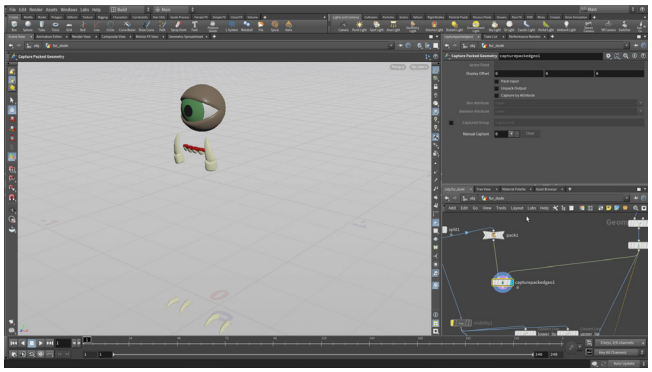




02 In the Network View, press **tab > Pack** and place the node down to the right of the *split* node. Wire the second output of the *split* node into the *pack* node and set its **Display Flag**. You can see the body parts in the Scene View.

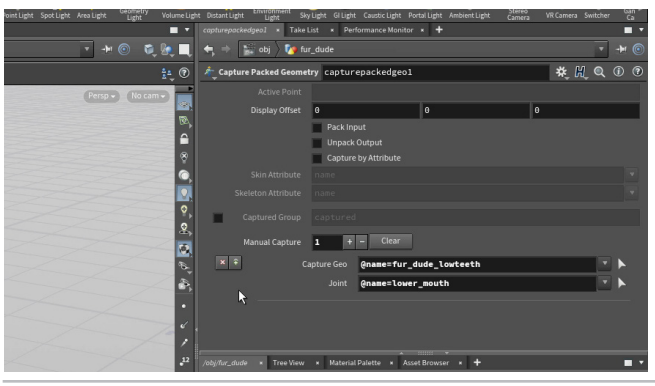
Turn the **Path Attribute** checkbox to **Off** then the **Name Attribute** checkbox to **On** and set **Transfer Attributes** to **name**.

Using the arrow in the top right of the Network view, choose **Split** In the **Geometry Spreadsheet,** you can see the 8 packed primitives. These can now be captured to the skeleton as rigid geometry.



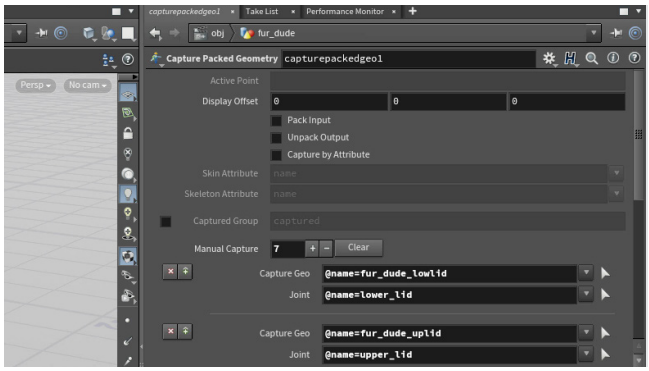
03 In the Network View, press **tab > Capture Packed Geometry** and place the node down under the *pack* node. Wire the output of the *pack* node into the first input of the *capturepackedgeo* node. Next wire the output of the skeletonmirror node into the second input of the *capturepackedgeo* node then set the Display flag on the *capturepackedgeo* node.

Nothing has happened yet - you need to associate the bones and the geometry you want to capture.



04 In the Parameter pane, click on the **plus sign** next to **Manual Capture**. Click on the **arrow** next to **Capture Geo** and in the Scene view, select the *lower teeth*. One click is all you need because the lower teeth are one of your packed groups. Press **Enter** to accept.

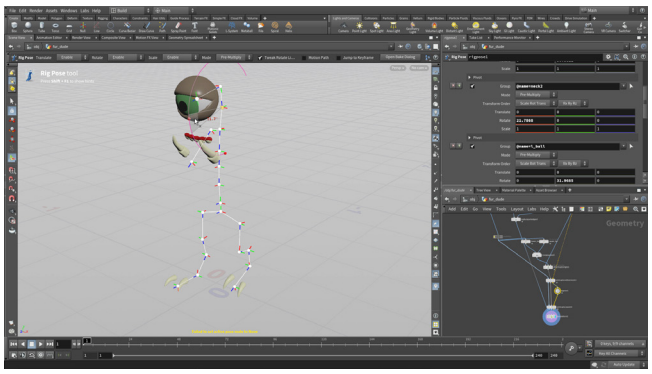
Now click on the **arrow** next to Joint. The geometry disappears and the skeleton is shown as lines and points. Select the *lower_mouth* joint and make sure your cursor is over the Scene View then press **Enter** to accept.



05 Click the plus sign and then the **arrow** next to **Capture Geo** and in the Scene view, shift select the *fur_dude_upteeth* and *fur_dude_gums* . Press **Enter** to accept. Now click on the **arrow** next to **Joint**. Select the *neck2* joint and press **Enter**.

Repeat these steps two more times to associate the following:

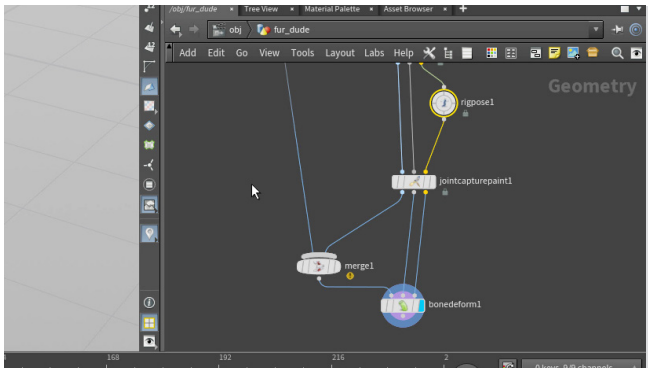
- *fur_dude_rclaws* to *r_toe*
- *fur_dude_lclaws* to *l_toe*
- *fur_dude_eye to eyeball*
- *fur_dude_uplid to upper_lid*
- *fur_dude_lowlid to lower_lid*



06 Wire the output of the *capturepackedgeo* **into the first input of the** *bonedeform* **node**. Set the Display flag on the *bonedeform* node.

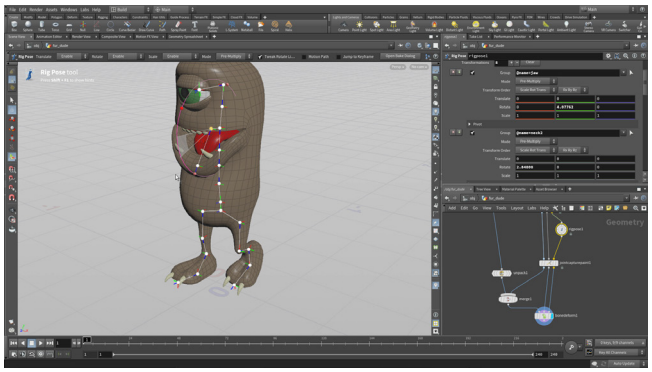
Select the *rigpose* node and then select and rotate joints in the Scene view. You can see that the geometry is being captured to the joints without any deformation.

The eye and eyelid joints overlap therefore when you click a little menu pops up letting you choose - you may need to click a few times to get the joint you need. You can then use them to rotate the eye and the eyelids separately.



07 In the Network View, press **tab > Merge** and place the node down in-between the *capturepackedgeo* and *bonedeform*. Wire the *capturelayerpaint* node into the *merge* node. On the *merge* node, press the blue up arrow next to *capturelayerpaint* to reorder the inputs.

Now everything is deforming but the body and tongue are grey while the packed geometry has its colors. There is also an error on the merge node because the one side has color (Cd) as a primitive attribute and the other side has it as a point attribute.



08 In the Network View, press **tab > Unpack** and place the node down between the *capturepackedgeo* and the *merge* nodes. Set **Iterations** to **2** and **Transfer Attributes** to * **^Cd**. The * grabs all the capture attributes and the **^Cd** make sure that you do not remove the original color attribute. Now both sides are using point colors and the rig looks correct.

Now you can select the *rigpose* node and then select and rotate joints in the Scene view. You can see that all the captured geometry is now working together.

Save your work.

FLATTENING THE NETWORK

The network that you have created to capture the geometry, paint weights and prepare the geometry for animation works well but can take a little time to update when changes are made.

To create a more efficient rig, you will flatten the network into a single file that has capture weights stored in the geometry. This file will deform efficiently when fed into a bone deform sop when accompanied by a skeleton with the same bones that were used to capture the geometry in the first place.

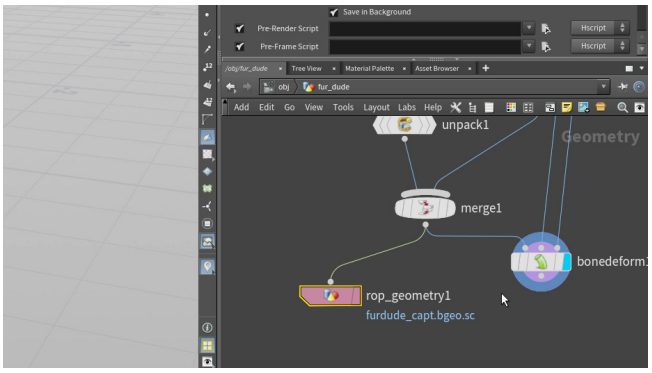
furdude_capt

furdude_skel

PART EIGHT

Create Capture Rig Digital Asset

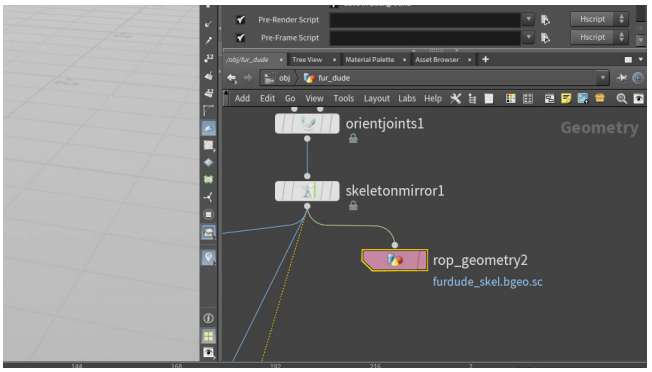
The captured geometry and the skeleton can now be wrapped up into a digital asset that can be used as the foundation for the animation rig. To start, you will export the geometry with its capture weights and the skeleton then embed these into the digital asset file. This will help make the capture rig more efficient when animating the character.



01 In the Network View, press **tab > ROP Geometry Output** and place the node down under the *merge* node. Wire the output of the *merge* node into the input of the *rop_geometry* node. In the Parameter pane, set **Output File** to:

$HIP/geo/furdude_capt.bgeo.sc

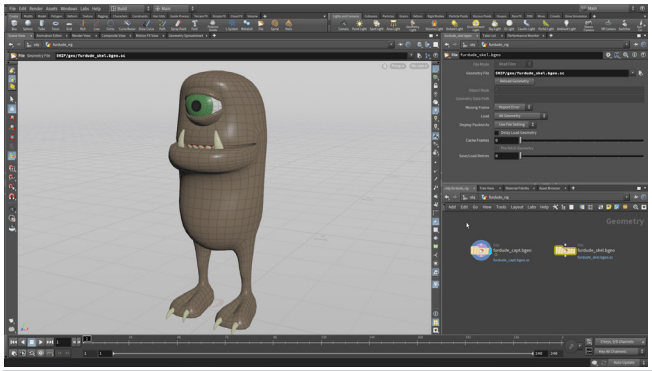
Click **Save to Disk** to save the geometry to your *geo* directory. This geometry looks like the geometry you imported at the beginning of the lesson but now it includes important data such as capture attributes that allow it to deform.



02 In the Network View, press **tab > ROP Geometry Output** and place the node down next to the *skeletonmirror* node. Wire the output of the *skeletonmirror* node into the input of the *rop_geometry* node. In the Parameter pane, set **Output File** to:

$HIP/geo/furdude_skel.bgeo.sc

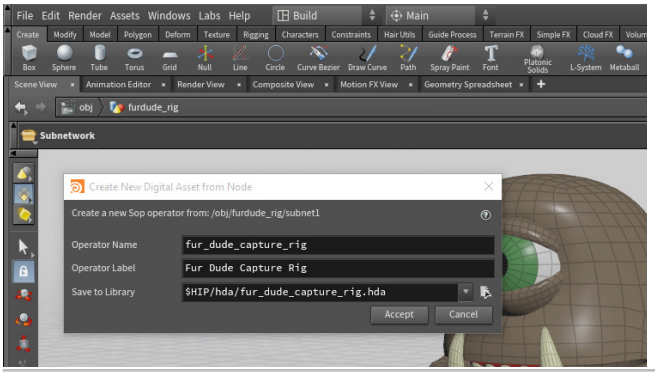
Click **Save to Disk** to save the geometry to your *geo* directory. This geometry represents your skeleton and can be used to build up an animation rig asset.



03 Go back up to the Object level. Rename the object to *fur_dude_capture*. Turn of it's **Display flag**.

Click on the **File** button on the **Create** shelf. Click on *$HIP* then go into the *geo* directory and select the *furdude_capt.bgeo.sc* file. Press **Enter** to place it at the origin then **rename** the new object node to *furdude_rig*. **Double click** to dive into this object. **Alt-drag** on the *File* node to create a second one. Set its **Geometry File** to: *$HIP/geo/furdude_skel.bgeo.sc*

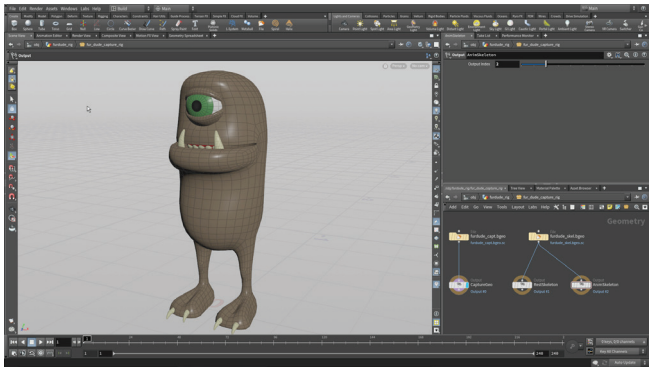
Rename it *furdude_skel.bgeo* then set the **Display Flag** on the *furdude_capt.bgeo* node.



04 Select the two *File* nodes then from the **Assets** menu select **New Digital Asset from Selection**. Set the following:

- **Operator Name** to *fur_dude_capture_rig*
- **Operator Label** to *Fur Dude Capture Rig*
- **Save to Library** to: *$HIP/hda/fur_dude.hda*

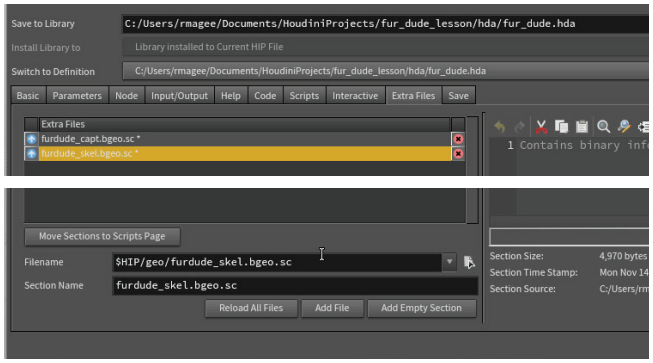
Click **Accept** to create the HDA file. An Edit Operator Type Properties window pops up. Set **Maximum Outputs** to **3** and click **Accept**. **Rename** the subnet to *fur_dude_capture_rig*.



05 **Double click** on this node to dive into the subnet. Press **tab > Output** and place an output node beneath the *fur_dude_geo_capt File* node. Wire the *File* node into the *output* node. Rename it *CaptureGeo*.

Alt-drag twice to create two new Output nodes. Name the second one *RestSkeleton* and set its **Output Index** to **1**. Name the third one *AnimSkeleton* and set its **Output Index** to **2**. Wire the *fur_dude_skel.bgeo File* node into both the second and third *output* nodes.

Set the **Display Flag** on the *CaptureGeo* output node.

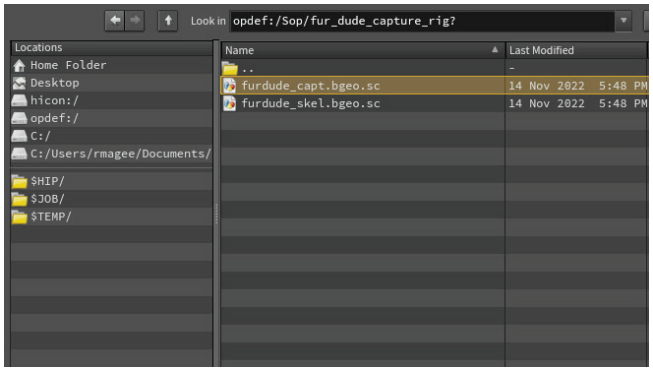


06 From the **Asset** menu, select **Edit Asset Properties > Fur Dude Capture Rig**. This opens up the Operator Type Properties window. Click on the **Extra Files** tab.

Click on the chooser button next to **Filename** in the lower left and navigate to *$HIP/geo/fur_dude_capt.bgeo.sc*. Click **Accept**. Then click **Add File**.

Repeat these steps for the *fur_dude_skel.bgeo.sc* file. Now you have placed these files inside the digital asset file which will make them easier to share with other people as a complete package.

Click **Accept** to finish.

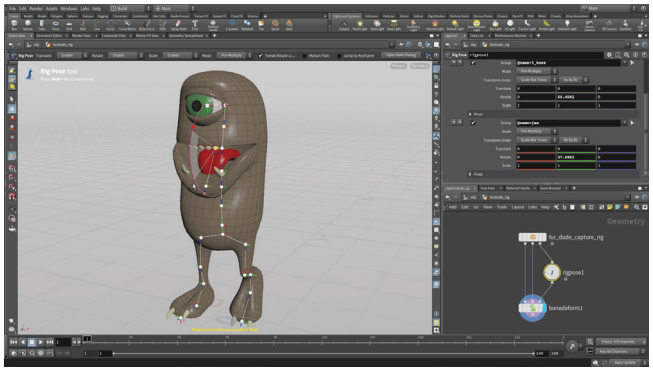


07 On the first file node, click on the **Chooser** icon next to **Geometry File** then click on *opdef:/* in the **Locations** sidebar then double click on the *sop* directory then again on the *fur_dude_capture_rig* folder. Select the *fur_dude_capt.bgeo.sc* file then press **Accept**. This creates the following opdef expression:

```
opdef:/Sop/fur_dude_capture_rig?furdude_capt.bgeo.sc
```

Repeat these steps for the *fur_dude_skel.bgeo.sc File* node.

From the **Assets** menu, select **Lock Asset > Fur Dude Capture Rig**. **Save Changes** to protect the contents of this asset. Later you can unlock it and update these files if needed.



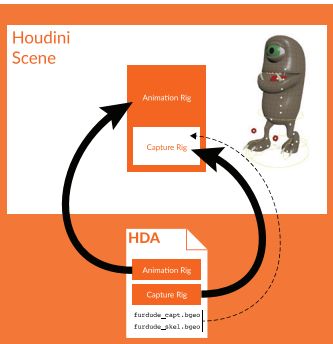
08 Go up one level where you will see the capture rig with three outputs. Press **tab > Bone Deform** and place this node underneath. Wire the three outputs of the *fur_dude_capt_rig* into the three inputs of the *bonedeform* node. Now add a rigpose node in the middle of the third chain. Set the **Display Flag** on the *bonedeform* node.

Select the *rigpose* node and make sure the **Handle** tool is active. You can again see all the skeleton joints. **Pose** the skeleton to test that the deformations are working in the same way as before.

HDA

This network of nodes was saved as a Houdini Digital Asset or HDA. This is a file on disk that is easy to share. The capture rig is an asset that is referenced off disk and in the next section, you will build the animation rig with the capture rig nested inside it. Both of them along with the bgeo files will be stored in a single HDA file.

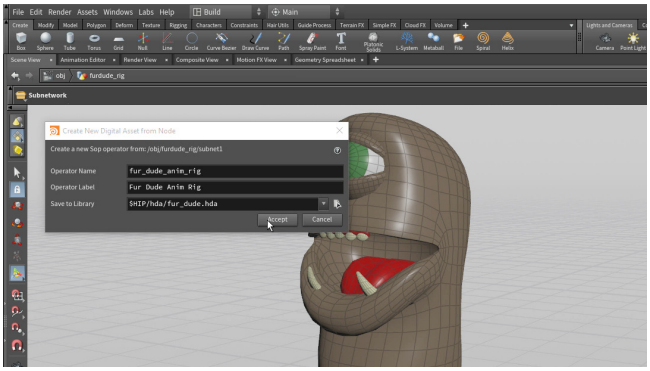
Changes made to the HDA file on disk will update instances of the asset being worked on by animators in their scene files.



PART NINE

Create the Animation Rig Asset

You are now going to create a second digital asset that has the capture rig nested inside it. This new asset will be the one that can be animated to create the final motion of the character. This new asset will contain all of the rigging tools such as inverse kinematics and aim constraints that assist with animation. In order to test these controls as you add them, you will set up a test version of the rig that is locked and visible in a second Scene view pane.



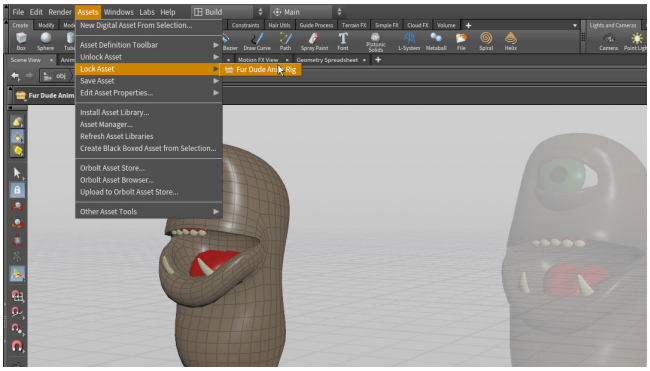
01

Select the three nodes then from the **Assets** menu select **New Digital Asset from Selection**. Set the following:

- **Operator Name** to *fur_dude_anim_rig*
- **Operator Label** to *Fur Dude Anim Rig*

For **Save to Library**, click on the browse button then click on $HIP the double click into the HDA directory. Select the fur_dude.hda file then click **Accept**. It is now set to: *$HIP/hda/fur_dude.hda*

Click **Accept** then **Accept** again in the Type Properties panel. This adds the new asset definition to the same HDA file. **Rename** the subnet to *fur_dude_anim_rig*.



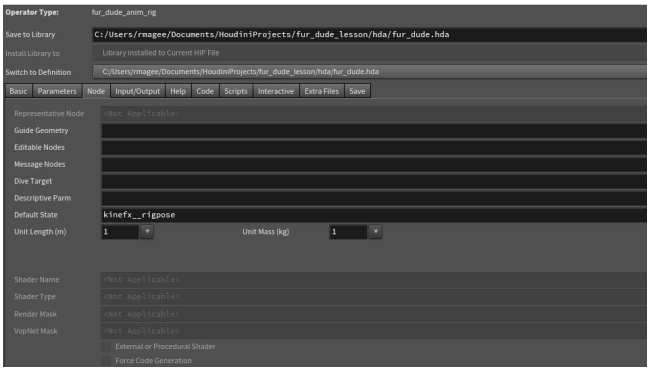
02

Go to the Object level. **Alt drag** on the *furdude_rig* node to create a new geometry node and name it *test_rig*. Press t and move the test rig over to the left.

Double click to dive into it. Select the *fur_dude_anim_rig* node and from the **Asset** menu, select **Lock Asset > Fur Dude Anim Rig.**

Go back up one level. You now have two versions of the rig - the *test_rig* is locked and shows you what it is like to interact with the completed asset. Right now you have nothing to interact with. You will fix that soon.

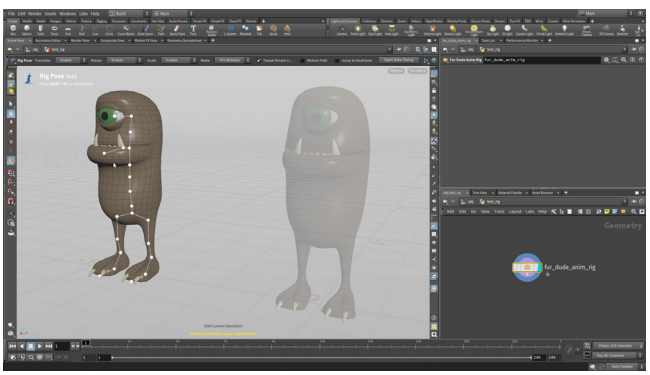
Click in the Network editor, and press **Ctrl 1** to set a quickmark.



03

Navigate back to the *furdude_rig* object then into *fur_dude_anim_rig*. Click in the Network editor, and press **Ctrl 2** to set a quickmark. Now you will be able to quickly jump back and forth between these networks.

Select *rigpose* then click on the **Clear** button next to **Transformations**. Go to the **Asset** menu, select **Save Asset > Fur Dude Anim Rig.** From the Asset menu, choose **Edit Asset Properties > Fur Dude Anim Rig**. Click on the **Node** tab and set **Default State** to: `kinefx__rigpose`. Click **Accept**.



04

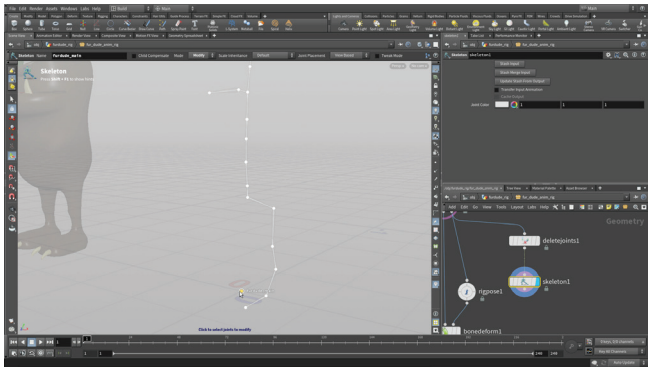
In the Network view, **press 1** to navigate back to the **test_rig**. In the Scene View on the left, expand the toolbar and click on the **Handle** tool.

You will now see joints displayed on the *test* rig. If you click on any of them, you will see that they can't be changed because their parameters have not been promoted to the asset. This is the beginning of building an animatable interface for this character.

PART TEN

Add More Control Joints

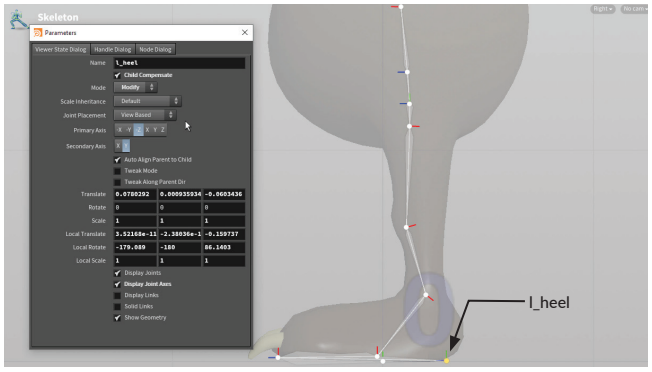
To provide more flexibility with the control rig, you can add joints such as a root joint for the whole skeleton, heel joints for a reverse foot setup or a look-at point for you to target with your eyeball. These joints will have the same names as the ones in the original rest skeleton and that will ensure that they are used to drive the motion on the character.



01 In the Network view, **press 2** to navigate back to the **fur_dude_anim_rig** and set its **Display flag**. Set the **Template flag** on the *fur_dude_capture_rig* node.

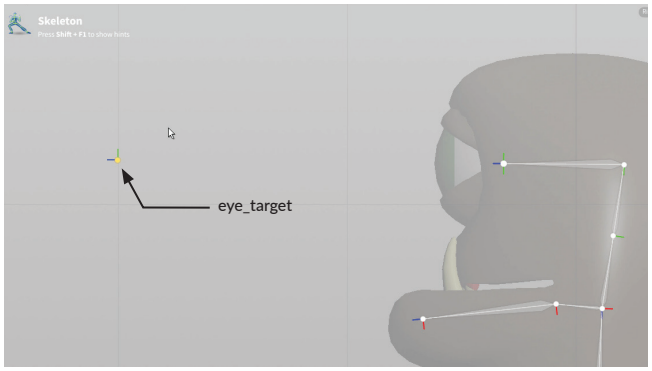
Branch off a **Delete Joints** node from the third output of the *fur_dude_capture_rig* node. Click on the arrow next to group then select fur dude's right leg joints. Press **Enter**.

Add a **Skeleton** node. Change to **Create Mode**. Turn on **Grid snapping** and add a point to the origin. Change back to **Modify** mode and rename the new joint to *furdude_main*.



02 Go to right view. Go back to **Create** model and **MMB-click** to break the connection to the main joint. Click on the *l_toe* joint then add a new joint where the heel would be.

Change back to **Modify** mode and rename the new joint to *l_heel*. Press **p** and with **Child Compensate** set to **On**, set the **Rotate** to **0, 0, 0**.



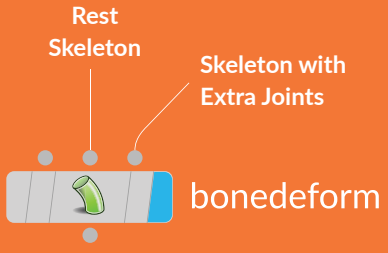
03 Pan to the area around the eye. Go back to **Create** model and **MMB-click** to any connections. Now add a new joint out in front of the eye.

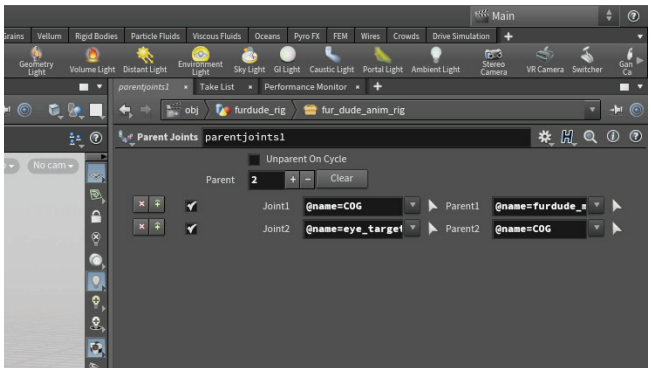
Change back to **Modify** mode and rename the new joint to *eye_target*. Press **p** and make sure that **Rotate** is set to **0, 0, 0**.

ADDING EXTRA JOINTS TO THE RIG

These extra joints were added to the stream of nodes on the far right while the middle rest skeleton will continue with the original joints. When fed into the bone deform, the extra "phantom" joints are ignored and only the original joints determine the final output of the rig.

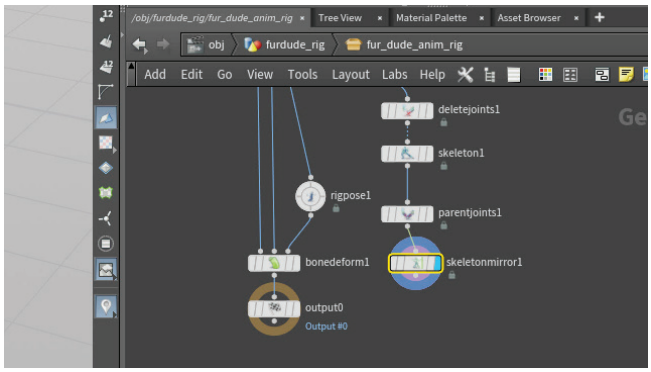
If you were to feed these into the middle rest skeleton input on bone deform you would get an error because these extra joints would not have corresponding capture weights in the incoming geometry.





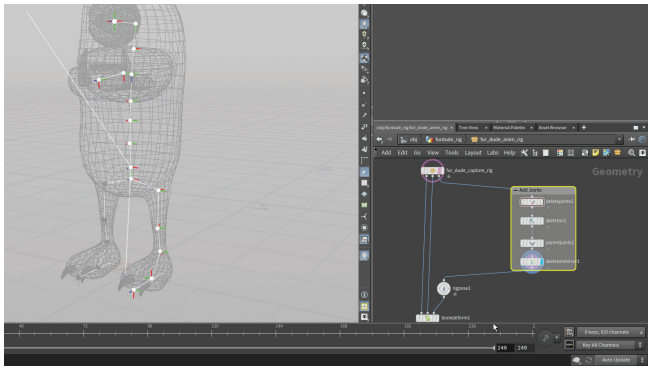
04 Add a **Parent Joints** node. Click the **plus sign** twice. Click on the arrow next to **Joint1** and in the scene view click on the *COG* joint. Press **Enter** (with your cursor over the Scene View) to accept. Now use the arrow next to **Parent1** and select the *furdude_main* joint.

For the second entry, click on the arrow next to **Joint2** and in the scene view click on the *eye_target* joint. Press **Enter** (with your cursor over the Scene View) to accept. Now use the arrow next to **Parent2** and select the *COG* joint.



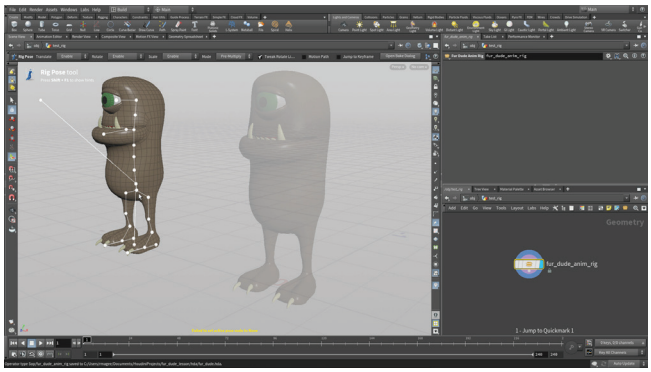
05 Add a **Skeleton Mirror** node to the chain. This creates a mirror copy of all the joints. Go to the Parameter pane and click on the arrow next to **Group**. Select only the left leg joints, including the new *l_heel* joint,and press **Enter**. Now only the leg is being mirrored.

Under naming, set **Find Tokens** to *l_* and **Replace Tokens** with *r_*. Now you have the right leg joints properly named.



06 Feed the skeleton mirror node into the *rigpose* node. Set the **Display flag** on the *bonedeform* node. The claws on the foot appear to have flipped. Go back to the skeleton joint, select the *l_toe* and **press p** to bring up the parameters. Set **Rotate** to **0, 0, -90**.

Select the four nodes you used to add joints and click on the **Add Network box** button. Position the box and center the nodes. Click on the box's title and enter *Add Joints*.



07 From the **Assets** menu, select **Save Asset > Fur Dude Anim Rig**. This saves the changes into the asset definition which will update on the *test_rig*. You still can't do anything with the *test_rig* because there are no parameters promoted to the top level.

You are now going to setup the main controls and promote parameters to start bringing the *test_rig* to life.

WHAT IS THE ROLE OF THE TEST RIG?

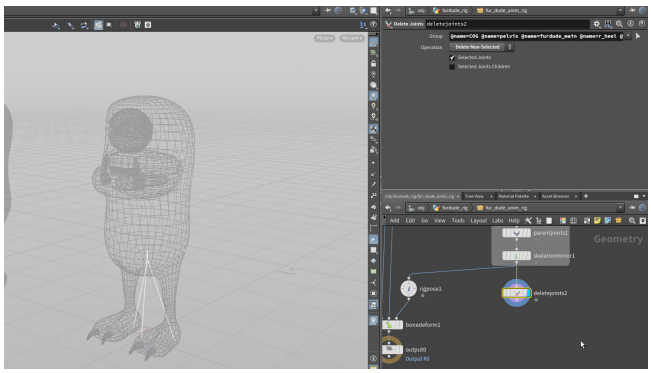
The rig you are working on is an unlocked asset which gives you the ability to work with all the joints inside the asset. But when the asset is published to an animator, they can only work with parameters that have been promoted to the top level of the character.

The *test_rig* is a second locked version of the asset and until you start promoting parameters and building controls, you will not be able to manipulate it. That is what makes it a great tool for verifying that the asset is ready for animation - if you can't work with the test rig then the animator will not be able to pose the character.

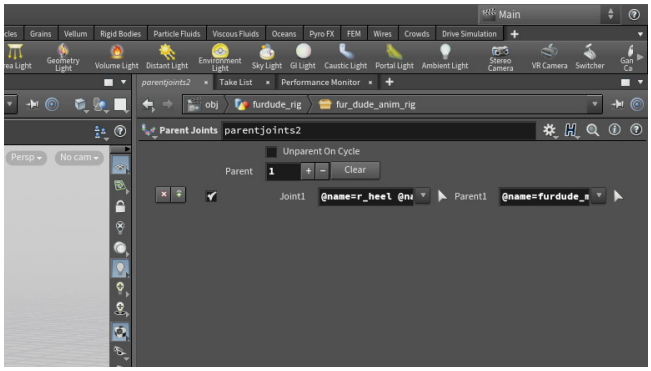
PART ELEVEN

The Main Controls

To add kinematics, you need to break the current hierarchy where the feet are under the COG. You can break off a few joints and reparent them to build the hierarchy you need. This reparenting happens off to the side then you will blend the results back into the original skeleton hierarchy which is important to make sure the bone deform functions properly.



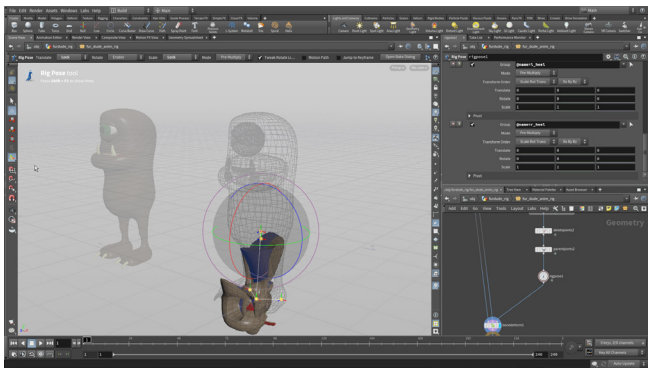
01 Branch off a **Delete joints** node from the *skeletonmirror* node. Click on the arrow beside Group and in the scene view select the *furdude_main*, the *COG*, the *pelvis*, the *l_heel* and *r_heel* joints. Press **Enter** then set **Operation** to **Delete Non-Selected**.



02 Add a **Parent Joints** node after the *deletejoints* node. Click on the **+ sign** to add a joint listing. Click on the arrow next to **Joint1** and in the scene view select the two *heel* joints. Press **Enter**.

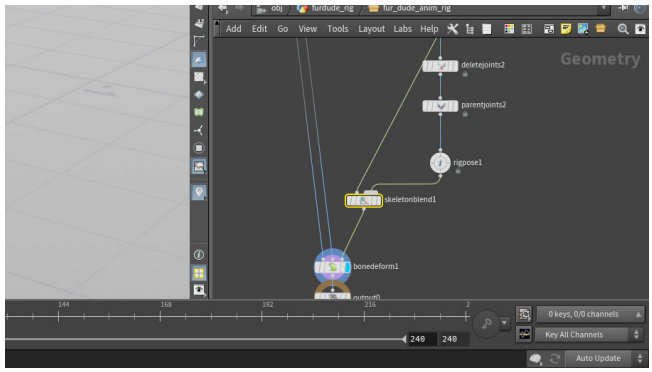
Click on the arrow next to **Parent1** and in the scene view select the *furdude_main* joint. Press **Enter**.

Now the heel joints are free from the *COG* and can be used to drive the IK on the legs down the line. But all of the joints are parented to the *furdude_main* joint except for the *pelvis* joint which is parented to the *COG* joint.



03 Wire the *parentjoints* node into the *rigpose* node. Clear any existing joints from this node then set the **Display flag** on it. Now click on the *furdude_main* joint, the *COG* joint, the *pelvis* joint then the two *heel* joints.

If you now set the **Display flag** on the *bonedeform* node everything is mixed up because most of the joints are now missing. This can be fixed with a skeleton blend.



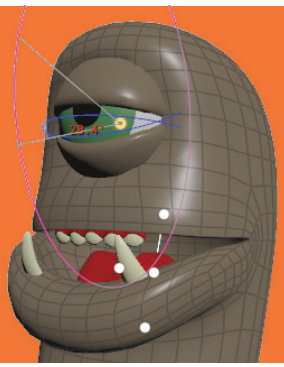
04 Add a **Skeleton Blend** node into the Network editor. Wire the *skeletonmirror* node into the left input and the *rigpose* into the right input. Then wire the *skeletonblend* node into the third input of the *bonedeform* node.

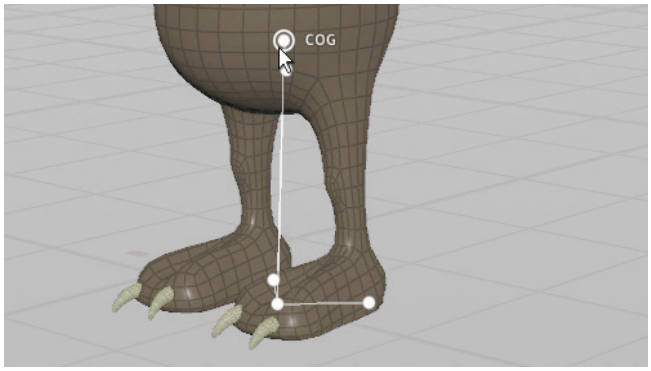
In the *skeletonblend* Parameter pane, set the **World Space** checkbox to **On** and *weight1* to **1**.

CONTROL GEOMETRY

When you add a joint to a rigpose, it can be promoted to the top level of the asset to select and animate. The **Attach Control Geometry** node lets you assign geometry you build to different parts of the rig to make it easier to select joints for manipulation. You can create any shape you want for these controls.

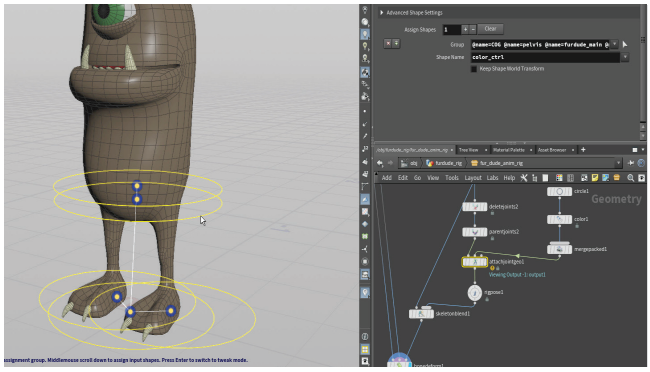
A good example where this will help is the eyeball and the two lid joints which overlap. Control Geometry will make it easier to select these when you set up that part of the rig. For now you will use it for the main controls.





05 From the **Assets** menu, select **Save Asset > Fur Dude Anim Rig**. This saves the current setup. In the Network view, **Press 1** to navigate back to the **test_rig** where you can see that only the five joints listed in the rigpose are visible.

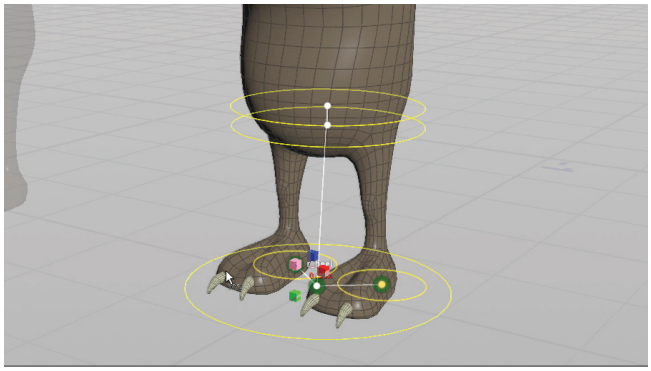
You cannot select and move them because the parameters haven't been promoted yet.



06 In the Network view, **press 2** to navigate back to the **fur_dude_anim_rig**. Add a **Circle** node to the network. Set **Orientation** to **XZ**. Set **Uniform Scale** to **0.2**. Set **Divisions** to **36** and **Arc Type** to **Open**. Add a **Color** node and set **Color** to **yellow**.

Follow with a **Merge Packed** node and set **Name 1** to *circle_ctrl*.

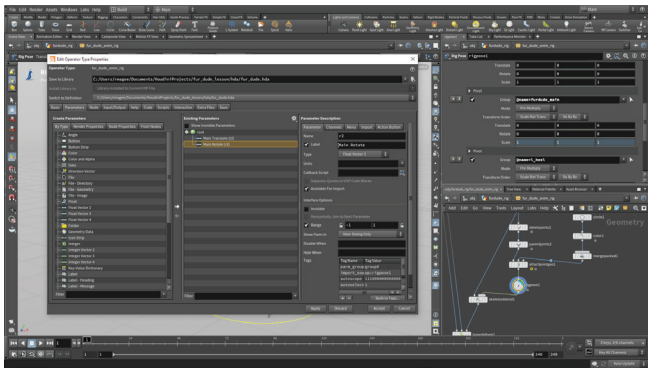
Place an **Attach Joint Geometry** node between the *parent* node and the *rigpose* node. Wire the *mergepacked* node into the second input. Set the **Display Flag** on this node and press **Enter** in the scene view. Select all the visible joints then **press G** and use your scroll wheel to find the *circle_ctrl* geometry. It gets assigned to all the joints.



07 In the Operation Control bar at the top, change the **Mode** to **Tweak Shapes**.

Select the *COG* and *Pelvis* joints then **press G** to bring up a transform handle. **Press E** to get the scale handle then click drag on the middle handle to scale in all three directions until these controls are a bit smaller (around 0.67 in the parameter pane).

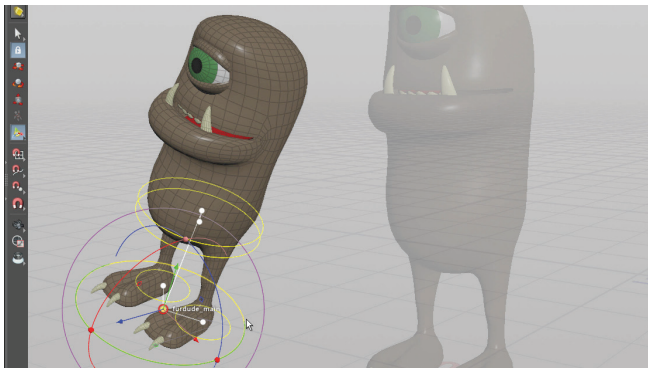
Select the two *heel* joints then **press G** to bring up a transform handle. **Press E** to get the scale handle then click drag on the middle handle to scale in all three directions until the heel controls are much smaller (around 0.3 in the parameter pane).



08 Select the *rigpose* node then select the four joints using the control geometry. To make this work on the test rig you need to promote parameters.

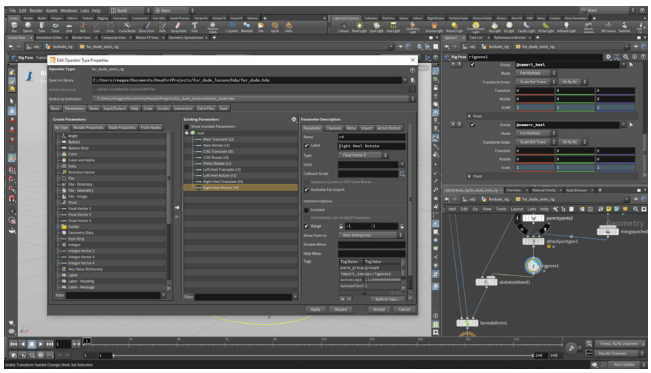
From the **Assets** menu, select **Edit Asset Properties > Fur Dude Anim Rig**. Click on the **Parameters** tab.

On the *rigpose* node go to *furdude_main*, **RMB-click** on **Scale** and choose **Lock Parameter**. Now drag the on **Translate** and move it over to the Parameter list under *root*. Set it's **Label** to *Main Translate*. Repeat for the **Rotate** parameters and name them *Main Rotate*. Click **Accept** to finish and save the results to the asset.



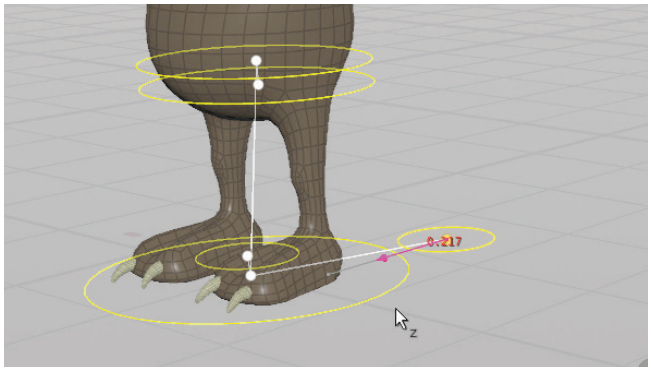
09 In the Network view, **press 1** to navigate back to the **test_rig**. The *test rig* now updates to show the new controls - as long as you have the **Handle** tool active. Select the *furdude_main* joint using its control geometry and now a transform handle appears that you can use to move the rig around.

Undo when you finish to put it back at the starting point..



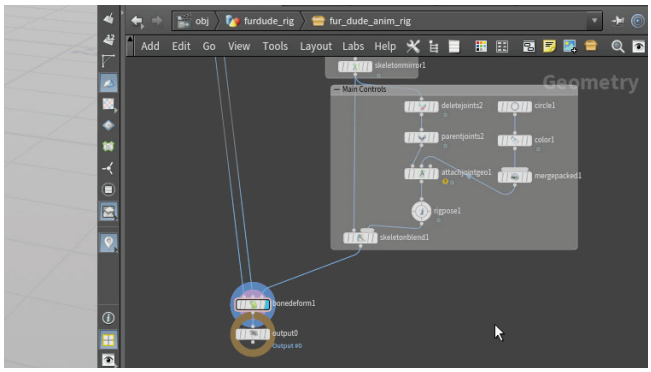
10 In the Network view, **press 2** to navigate back to the **fur_dude_anim_rig**. From the **Assets** menu, select **Edit Asset Properties > Fur Dude Anim Rig**. Drag the **Translate** and **Rotate** for the *COG*, *l_heel* and *r_heel joints* and the **Rotate** for the *pelvis*. In each case, lock the unused **Scale** or **Translate** (for the pelvis) parameters for each joint.

Click **Accept** to finish and **Save** the results to the asset.



11 In the Network view, **press 1** to navigate back to the **test_rig** which has been updated to show the new controls. Select the *COG* and *heel* joints using the control geometry and transform the parts around.

Undo when you finish to put all the parts back to their original location.



12 In the Network view, **press 2** to navigate back to the **fur_dude_anim_rig**. Select the nodes you used to set up the main controls and click on the **Add Network box** button. Position the box and center the nodes. Click on the box's title and enter *Main Controls*.

From the **Assets** menu, select **Save Asset > Fur Dude Anim Rig**. This has no effect on the rig but keeps the asset up-to-date. You may want to **Save** your Scene file too.

ORGANIZING YOUR NETWORK

Lining up nodes and adding **network boxes** are extra steps that are worth the added effort. The more organized your network is, the easier it is for you to work with later or for others to read what your intentions are.

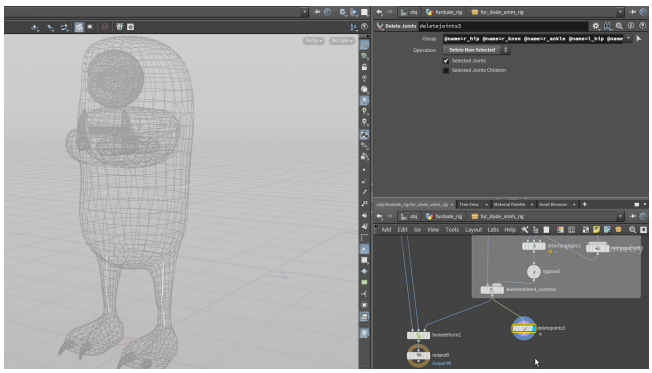
You can also add comments to each node and display them in the network or you can use **sticky notes** to explain larger blocks of nodes. Communication is always beneficial when creating networks in a team setting.

This part of the network organizes the main controls such as the root, the COG and the heel joints.

PART TWELVE

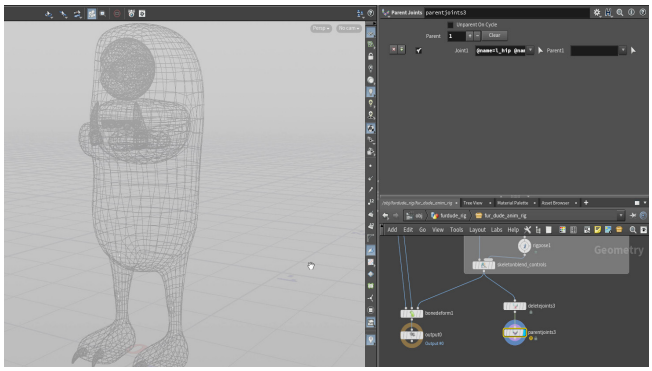
Inverse Kinematics for the Legs

To animate a character, Inverse Kinematics allow you to set up the legs so that moving either the feet or the hips causes the knee to bend appropriately. You are again going to pull out some joints from the main skeleton and set them up using KineFX. You will again blend the results back into the original hierarchy.

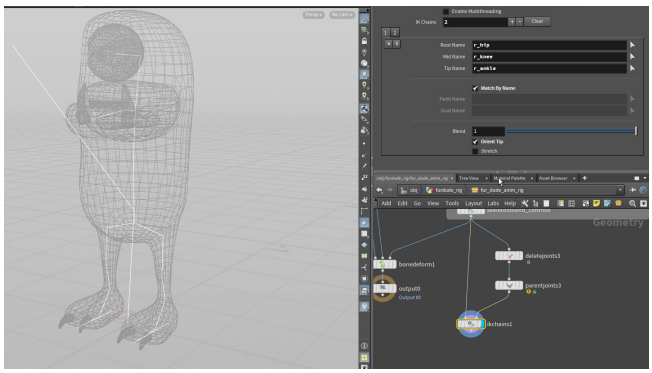


01 Rename the *skeletonblend* node to *skeletonblend_controls*. You are going to use these nodes a lot and it will be helpful to be able to identify them.

Branch off a **Delete joints** node from the *skeletonblend_controls* node and set its **Display flag**. Click on the arrow beside Group and in the scene view select the left and right *hip, knee and ankle* joints. Press **Enter** then set **Operation** to **Delete Non-Selected**.



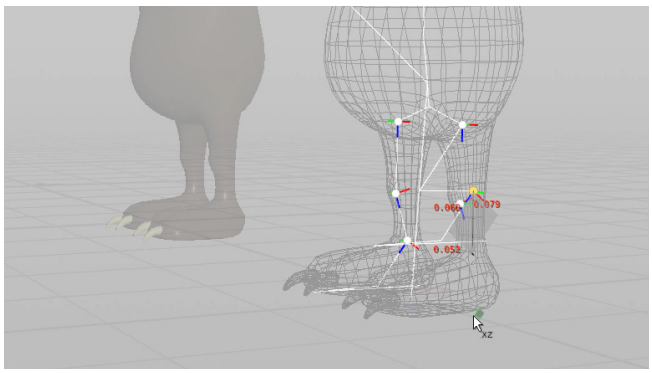
02 Branch a **Parent Joints** node and set its **Display flag**. Click the **+ sign** to add a Joint and set **Joint1** to *. Leave **Parent1** left blank. Now all of the joints are unconnected. This will leave you free to use them independently.



03 Place an **IK Chains** node into the Network editor. Feed the *skeletonblend_controls* into the first input and the *parentjoints* node into the second one. Set its **Display Flag**.

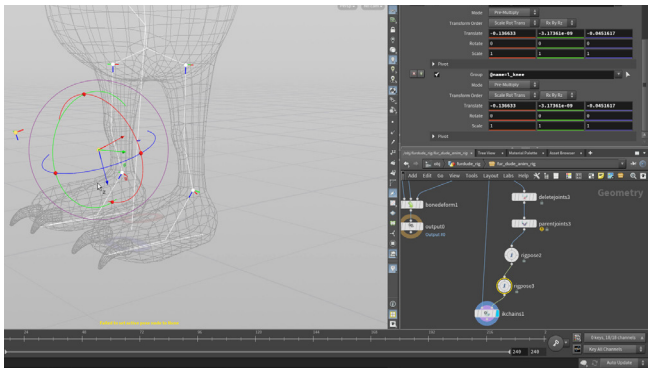
In the Parameter pane, click on the **+ sign** and in the folder click on the arrow next to **Root name**. Select the *l_hip* joint and press Enter (with your cursor over the Scene View). Set the **Mid Name** to *l_knee* and the **Tip Name** to *l_ankle*. You can use the arrow to select the joints or just type in the name. Set **Match by Name** to **On** and **Blend** to **1** then set **Orient Tip** to **On**.

Click the **+ sign** again and do the same for the right leg.



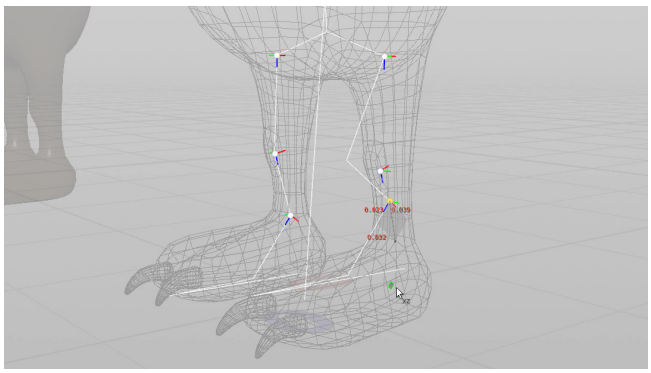
04 Add a **Rig Pose** node in-between the *parent* node and the *ikchains* node. Click on the *ankle* joints and move them around to see the Inverse Kinematics at work. Because you used **Match by Name** the ankle joints are acting as the end effectors for the IK chain.

You will see some flipping as you move the ankles. This is because the knee joints are being used as the twist effectors and they are not positioned very well - you need to move them in front of the legs.

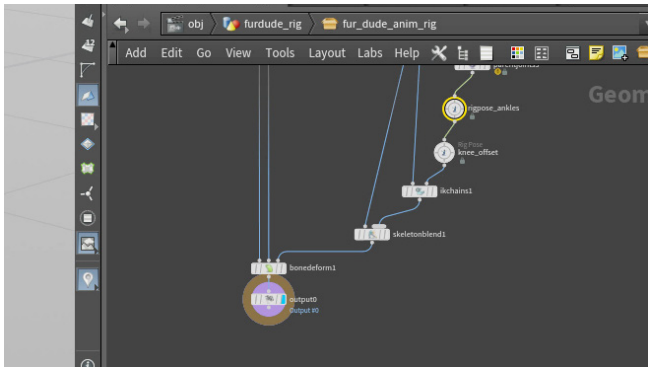


05 Add another **Rig Pose** node in-between the first *rigpose* node and the *ikchains* node. Press the **Shift** key and select both the *knee* joints and move them in front of the character. Now any flipping you had in your IK will have flipped back.

Name the first rigpose node to *rigpose_ankles* and the second one to *knee_offset*.

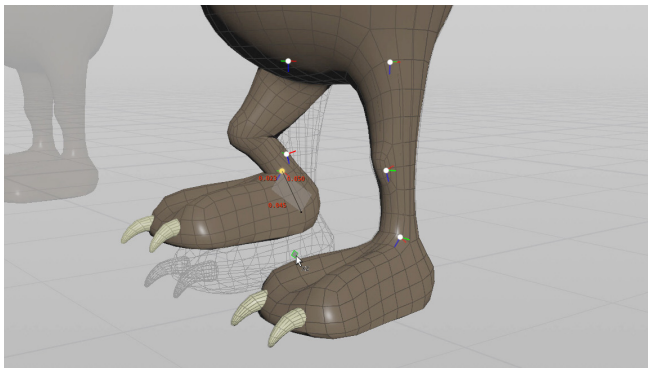


06 You can now go back to the *rig_ankles* node and test out the ankle joints. They don't flip now that you have the knees offset. You can also go to the *knee_offset* node and move the knees to act as a twist effector for the IK chain.



07 Add **Skeleton Blend** node and wire the *skeletonblend_controls* node into the first input and the *ikchains* into the second input. Then plug the output of the new **Skeleton Blend** node into the third input of the bone deform.

Rename the node to *skeletonblend_ik* and set the **World Space** checkbox to **On** and *weight1* to **1**.

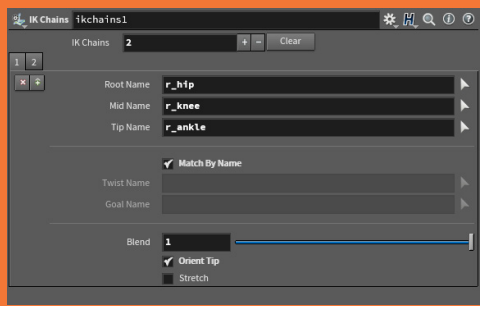


08 You can now go back to the *rigpose_ankles* node and test out the ankle joints to see how it affects the captured surface.

At this point, you are not going to save the digital asset to push these changes to the test rig. The *rigpose_ankles* node will not be how you control the ankles in the final rig. You are now going to build a reverse foot setup that will allow you to drive the whole foot setup and then the whole leg setup will be saved to the asset.

FK/IK BLENDING

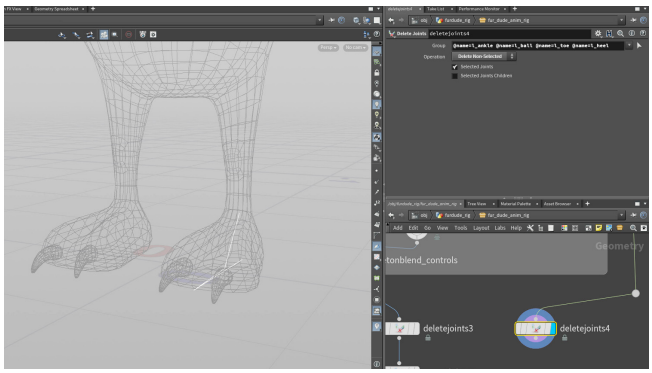
There are a couple of ways to blend IK and FK but you wont' be using them in this lesson because you will only be using IK for the legs. To get blending to work, you would need to promote parameters for the leg joints to the asset using a rigpose then you could either use the **Blend** attribute on the IK Chains node or use a **Skeleton Blend** between the IK solution and the rotating joints in the rigpose. If Fur Dude had arms then it would make more sense to set this up.



PART THIRTEEN

Reverse Foot Setup

To control the feet, you will create a classic reverse foot setup where the heel becomes the root then the toe, ball and ankle are parented to that. This can be easily accomplished in KineFX and the results blended back into the original skeleton. In this case you will completely rebuild the right foot but since the joint names align everything works properly.

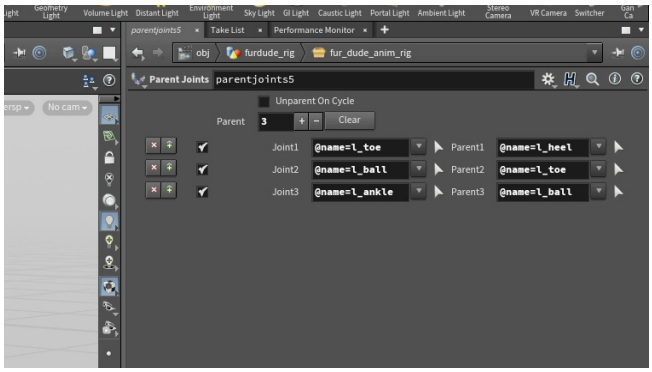


01

Delete the *rigpose_ankles* node. You will control the ankles using a reverse foot setup.

Branch off a **Delete joints** node from the *skeletonmirror* node and set its **Display flag**. Click on the arrow beside Group and in the scene view select the *l_ankle*, *l_ball*, *l_toe* and *l_heel* joints. Press **Enter** then set **Operation** to **Delete Non-Selected**.

Note: If you want redirect the connecting wire to go around the network box then you can **Alt-click** on it to add dots.

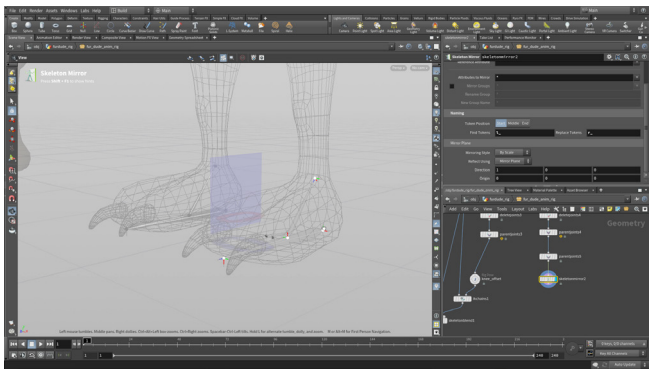


02

Add a **Parent Joints** node and set its **Display flag**. Click the **+ sign** to add a Joint and set **Joint1** to *. Leave **Parent1** left blank.

Branch a second **Parent Joints** node and set its **Display flag**. In the Scene View click on the *l_heel* joint then the *l_toe* joint then the *l_ball* joint then the *l_ankle* joint. MMB-click to finish. In the Parameter pane they will be displayed in the following order:

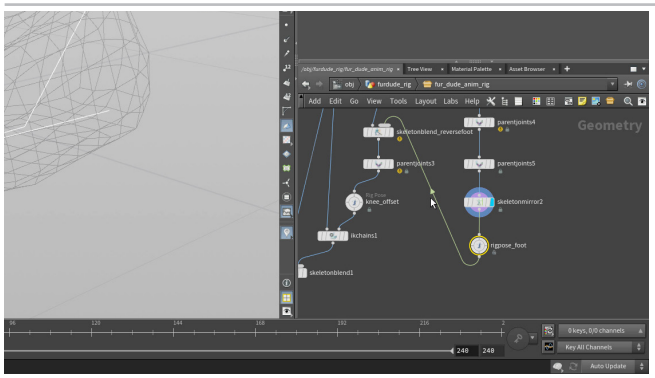
- **Joint1** to *@name=l_toe* | **Parent1** to *@name=l_heel*
- **Joint2** to *@name=l_ball* | **Parent2** to *@name=l_toe*
- **Joint3** to *@name=l_ankle* | **Parent3** to *@name=l_ball*



03

Add a **Skeleton Mirror** node to the chain. Go to the Parameter pane and under naming, set **Find Tokens** to *l_* and **Replace Tokens** with *r_*. This creates us the reverse foot for the right leg.

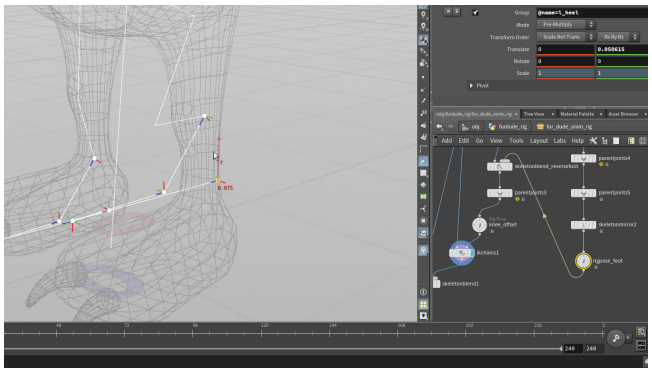
Because all the joints in this hierarchy have the same names as the original skeleton it will transfer the information properly when this rig is posed.



04

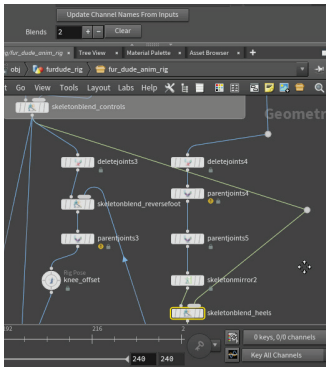
Now add a **Skeleton Blend** between the *deletejoints* and *parentjoints* that you set up earlier for the leg. Rename the node to *skeletonblend_reversefoot* and set the **World Space** checkbox to **On** and *weight1* to **1**. Click on the arrow next to **Group** and select the two *ankle* joints. Press **Enter**.

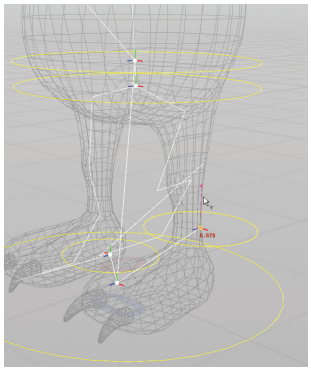
Add a **Rig Pose** node after the *skeletonmirror*. Rename it *rigpose_foot*. Feed it into the second input of the *skeletonblend_reversefoot* node.



05 Set the **Display flag** on the *ikchains* node. Now go to the *rigpose_foot* node and select the *l_heel* joint. Now when you move it you are moving the whole foot and the leg chain. Select the *l_toe* joint and rotate it. Again the reverse foot works and the IK chain is activated.

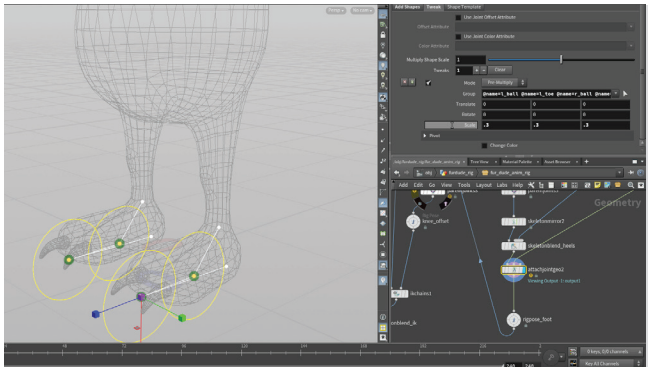
When you finish press **Clear** to remove all the joints. You will add some of them back later. For the heel joints, you want to use the heel joints you set up earlier as part of the main controls.





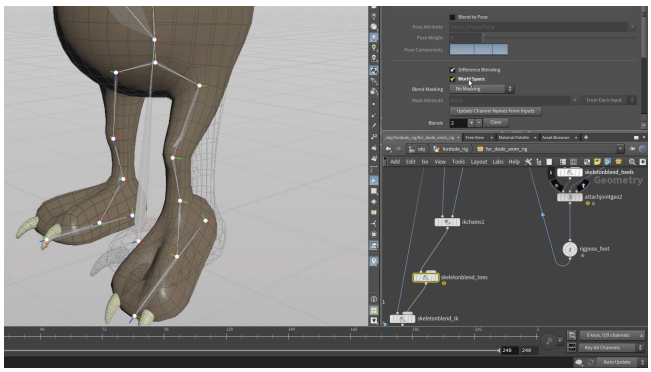
06 Now add a **Skeleton Blend** between the reverse foot *skeletonmirror* and *rigpose* node you were just using. Rename the node to *skeletonblend_heels* and set the **World Space** checkbox to **On** and *weight1* to **1**. Now feed the output of the *skeletonblend_controls* node into the second input.

Click on the arrow next to **Group** and select the two *heel* joints. Press **Enter**. Now you can select the *rigpose* for the controls and move the left or right heel to control the whole setup. You can also grab the COG joint and if you move it up and down the IK chain works properly.



07 Put an **Attach Joint Geometry** node between the *skeletonblend_heels* node and the *rigpose_foot* node. Wire the *mergepacked* node from the **Main Controls** network box into the second input. Set the **Display Flag** on this node and press **Enter** in the scene view.

Select the *toe* and *ball* joints then **press G** and use your scroll wheel to find the *circle_ctrl* geometry. In the Operation Control bar at the top, change the **Mode** to **Tweak Shapes**. Select the *toe* and *ball* joints then **press G**. **Press E** to get the **Scale** handle then click drag on the middle handle to scale the controls to around **0.3**.

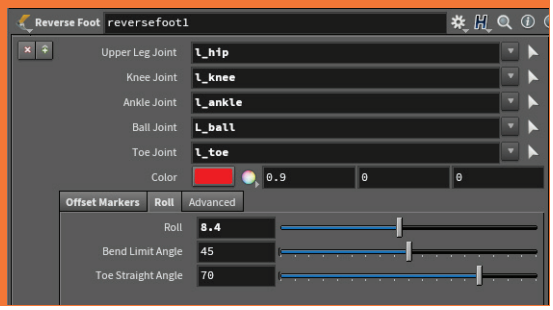


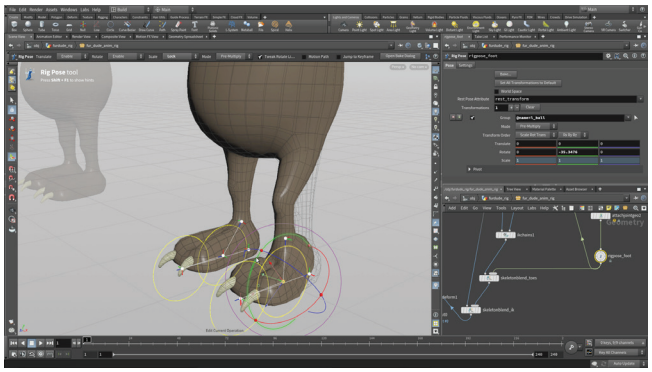
08 Set the **Display flag** on the *bonedeform* node. Pose the *l_ball* using the *rigpose_foot* node. You can see that rolling the ball causes the toe to point down instead of bending.

REVERSE FOOT SOP

There is a **Reverse Foot SOP** that you could use to control the feet and drive the leg kinematics. The **Reverse Foot SOP** has sliders for controlling the roll of the foot an individual controls for all the parts. But you are not using it in this lesson.

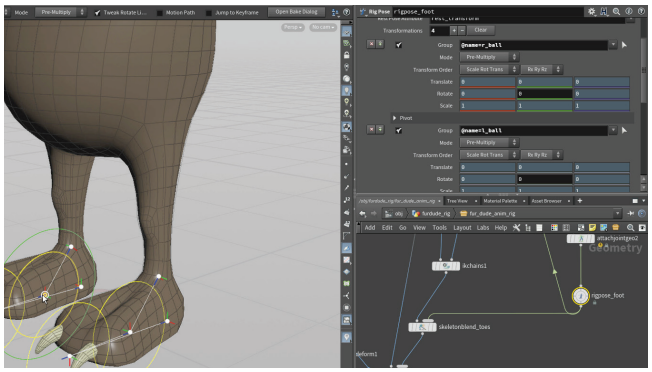
Instead you will create a hand-built reverse foot solution to learn how joints can be manipulated to give you the control you need.





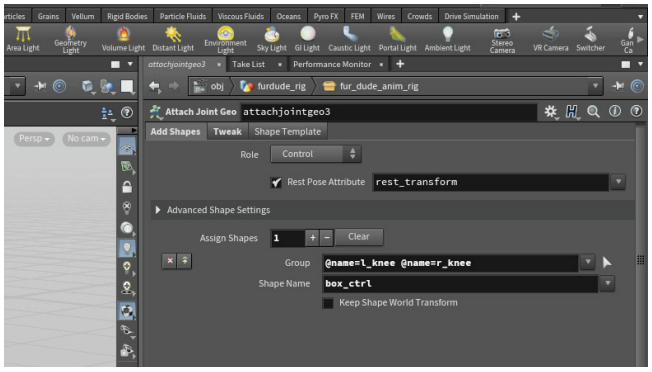
09 Now add a **Skeleton Blend** between the *ikchains* and *skeletonblend_ik* node you were just using. Rename the node to *skeletonblend_toes* and set the **World Space** checkbox to **On** and *weight1* to **1**. Now feed the output of the *rigpose_foot* node into the second input. Click on the arrow next to **Group** and select the *toe* and *ball* joints. Press **Enter**.

Now the toe will point in the right direction if you roll the *ball* joint.



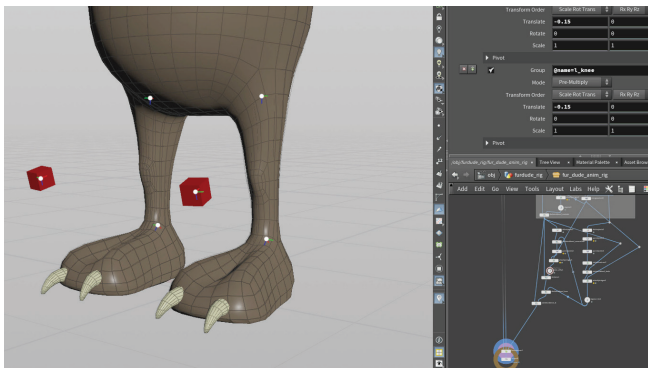
10 Select the *rigpose_feet* node. Make sure only the *left and right ball and toe* joints are listed on this node. For all of them, **RMB-click > Lock Parameter** on the **Translate, Rotate** and **Scale** parameters. Then go to the **Rotate Y** for each of them and **RMB-click > Unlock Parameter**.

Now if you select any of these four joints you will only get a Rotate Y handle.



11 Go back to the Main Controls Network box. Add a **Box** node to the network. Set its **Uniform Scale** to **0.02**. Follow this with a **Color** node and set it to a **red** color. Feed this node into the *mergepacked* node and set **Name 2** to *box_ctrl*.

Put an **Attach Joint Geometry** node between the *parentjoints* node and the *knee_offset* node. Wire the *mergepacked* node into the second input. Set **Mode** to **Tweak Shapes** then select the *knee* joints then **press G** and use your scroll wheel to find the *box* geometry.



12 Go to the *knee_offset* rigpose node. Make sure that only the two knee joints are listed. For both of them, **RMB-click > Lock Parameter** on the **Rotate** and **Scale** parameters. You will control these joints by moving them around.

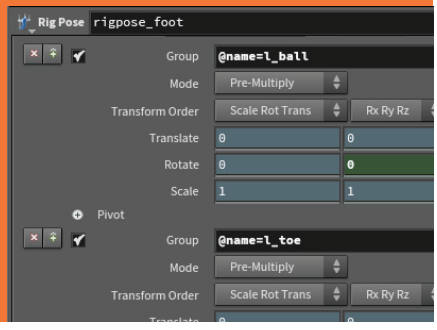
Set the **Translate** values to **-0.15, 0, -0.05** for both knees.

Add a **Network Box** around all of these nodes used to define the spine and head joints and name it *Leg Controls*.

THE ROLE OF RIG POSE

Up to this point the Rig Pose node has been used to test out the rig. When building an animation control rig, this node is also used to set up the parameters that will be promoted to the top level and define which joints will be visible when working with the digital asset.

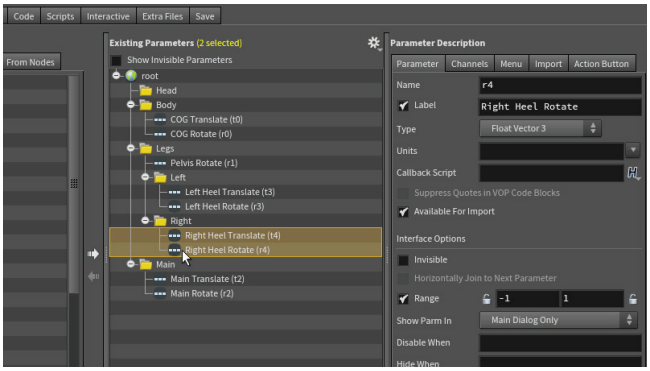
As you set up these nodes it is easy to accidentally add parameters that you don't need which will add extra joints at the top level. You can also make the mistake of deleting parameters that you do need by clicking on the x button.



PART FOURTEEN

Promote the Leg and Spine Controls

To make all of the leg and spine controls available to the animator, the parameters need to be promoted to the top level of the asset. This is an important step that is always needed to give the animator the control they need. This also means that you can keep certain parameters hidden that you don't want animators to work with.

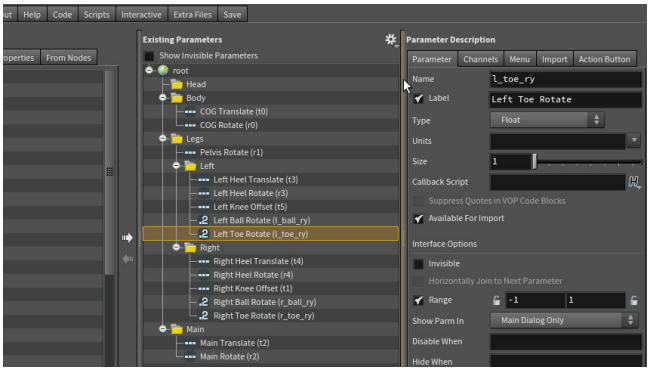


01

From the **Assets** menu, select **Edit Asset Properties > Fur Dude Anim Rig**. Click on the **Parameters** tab.

Create four folders. Name them *Head*, *Body*, *Legs* and *Main*. Drag the *COG parameters* onto the **Body** folder and the *Main parameters* onto the **Main** folder.

Put two folders inside the **Legs** folder and name the **Left** and **Right**. Add the *Left Heel* parameters into the **Left** folder and the *Right Heel* parameters into the **Right** Folder.

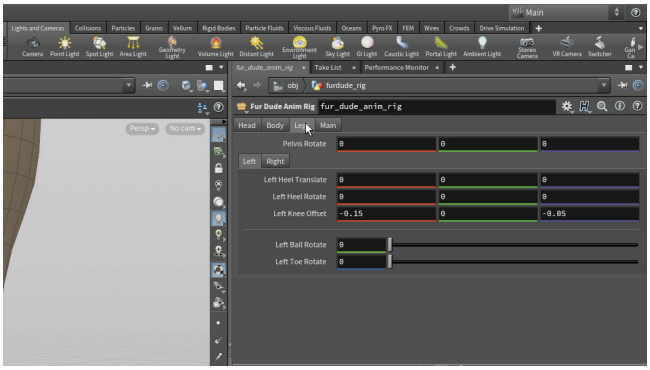


02

From the *knee_offset* rig pose node, drag the *l_knee* translate values into the **Left** folder in the **Leg** folder. Rename it **Left Knee Offset.** Click on the **Channel** tab and set the default to **-0.15, 0, -0.1.**

From the *rigpose_foot* node, drag the **Rotate Y** from the *l_ball* joint, name it **Left Ball Rotate** and set its **Range** to **0 to 30**. **Rotate Y** from the *l_toe* joint, name it **Left Toe Rotate** and set its **Range** to **0 to 20**. Drag a **Separator** in between the *Knee Offset* and *Ball Rotate* parameters.

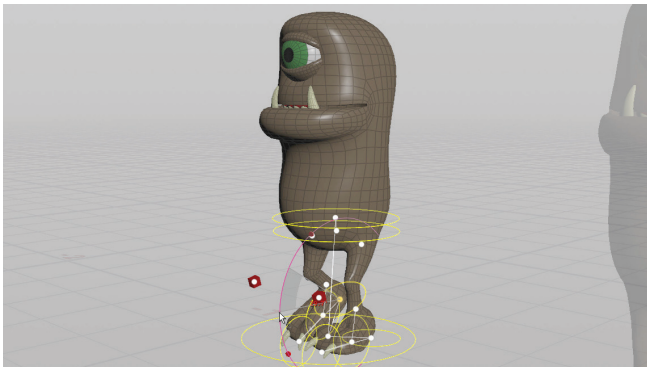
Repeat for the right knee and the right foot.



03

Click **Accept** and now the new parameters and controls are saved into the asset. Go up one level to see the asset parameters laid out on the *furdude_rig*.

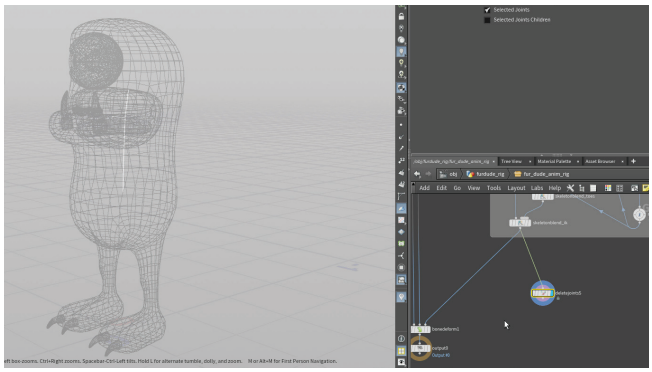
These parameters are for the unlocked rig that you are working on. Rather than test out these parameters, it is a better idea to play around with the *test_rig*.



04

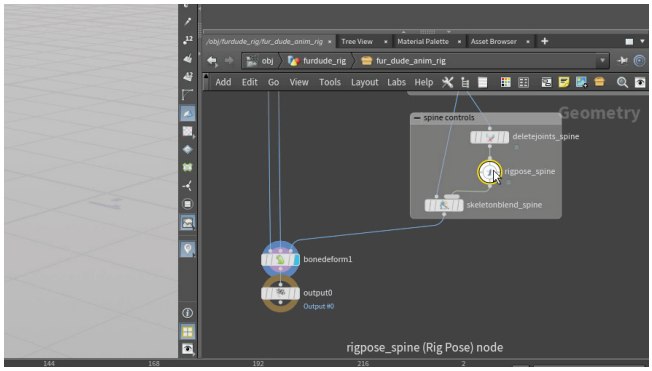
In the Network view, **press 1** to navigate back to the *test_rig* which has been updated to show the new controls.

Play with the test rig in the Scene view using the handles or using the parameters in the floating panel.



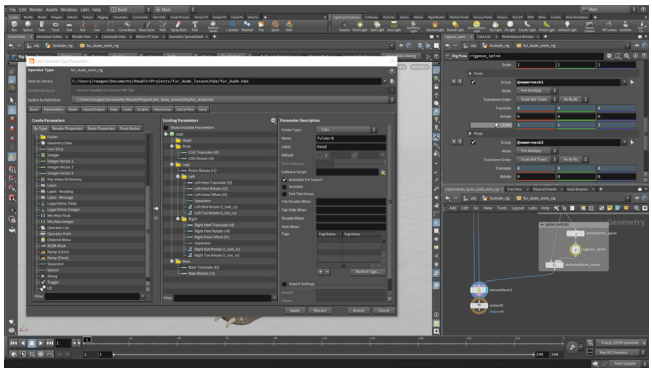
05 Branch off a **Delete joints** node from the *skeletonblend_ik* node and set its **Display flag**. Rename the node *deletejoints_spine*.

Click on the arrow beside **Group** and in the scene view select the *spine1*, *spine2*, *spine3*, *neck1*, *neck2* and *jaw* joints. Press **Enter** then set **Operation** to **Delete Non-Selected**.



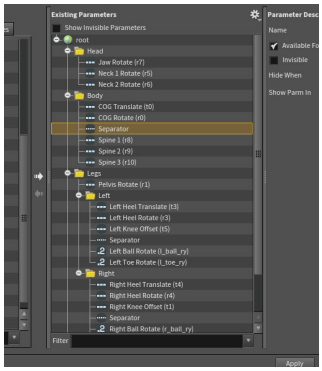
06 Add a **Rig Pose** node and rename it *rigpose_spine*. Now add a **Skeleton Blend** between the *skeletonblend_ik* and *bonedeform* node. Rename the node to *skeletonblend_spine* and set the **World Space** checkbox to **On** and *weight1* to **1**. Now feed the output of the *rigpose_spine* node into the second input.

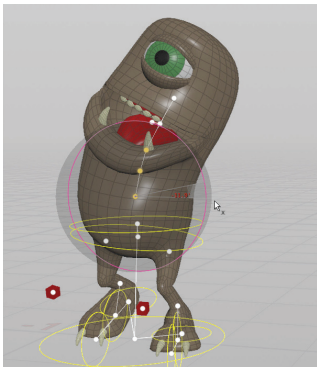
Add a **Network Box** around all of these nodes used to define the spine and head joints and name it *Spine Controls*.



07 Set its **Display Flag** then in the Scene view press and hold the **S key** then select all the joints. This will add them to the rigpose list.

Lock the **Translate** and **Scale** parameters for all of these joints. You will only be rotating them.





08 From the **Assets** menu, select **Edit Asset Properties > Fur Dude Anim Rig**. Click on the **Parameters** tab.

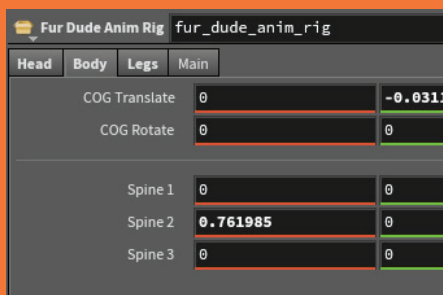
Drag the *neck1*, *neck2* and *jaw* onto the **Head** folder and name them Neck 1 Rotate, Neck 2 Rotate and Jaw Rotate. *spine1*, *spine2* and *spine3* onto the **Body** folder and name them *Spine 1*, *Spine 2* and *Spine 3*. Add a separator between the COG parameters and the spine parameters.

Click **Accept**. This will save the new controls to the rig. You can now explore them using the *test_rig*.

SPINE CONTROLS

For this rig, you set up the spine using joint rotations. This is known as **Forward Kinematics** and just like other parts of the rig, you pulled these joints out from the main skeleton then set them up in a Rig Pose to be promoted to the top level.

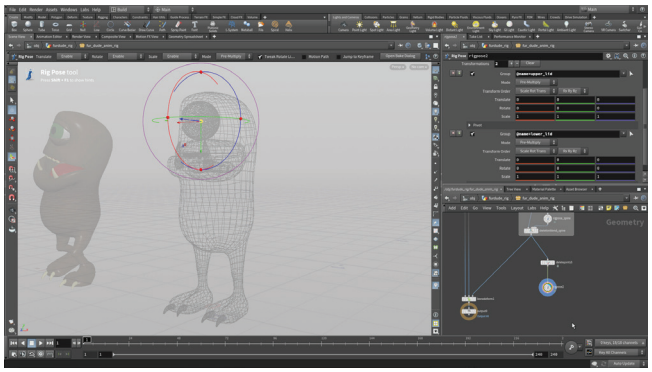
You did not use control geometry for these parts because the joints are easy to select in the Scene view. You don't need to have control geo for all the joints.



PART FIFTEEN

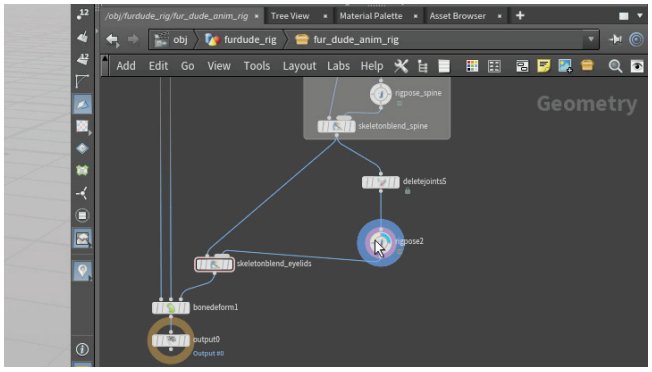
Eye Controls

The next step is to set up the eyelids with control geometry to make it easier to select these overlapping joints. You will also set up the eye target joint as a look at for the eyeball using a different section of Houdini called VOPS. When these parts are rigged, you will again promote the appropriate parameters to the character's asset.

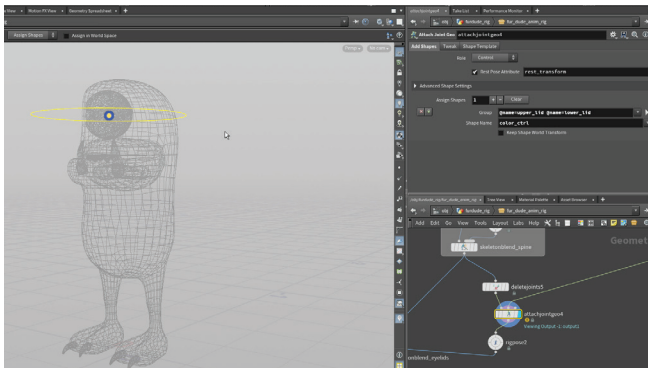


01 Branch off a **Delete joints** node from the *skeletonblend_spine* node, set its **Display flag** and name it *deletejoints_eyelids*. Click on the arrow beside **Group** and in the **Rig Tree** select *upper_lid and lower_lid*. Move your cursor over the Scene View and press **Enter** then set **Operation** to **Delete Non-Selected**.

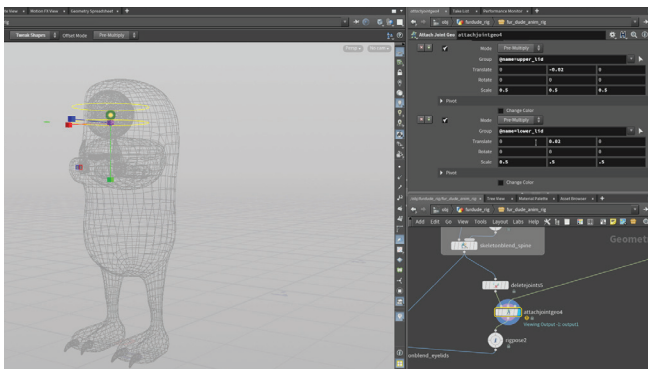
Add a **Rig Pose** node and rename it *rigpose_eyelids*. Set its **Display Flag** then in the Scene view press and hold the **S** key then select all the joints. This will add them to the rigpose list.



02 Now add a **Skeleton Blend** between the *skeletonblend_spine* and *bonedeform* node. Rename the node to *skeletonblend_eyelids* and set the **World Space** checkbox to **On** and *weight1* to **1**. Now feed the output of the *rigpose_eyelids* node into the second input.



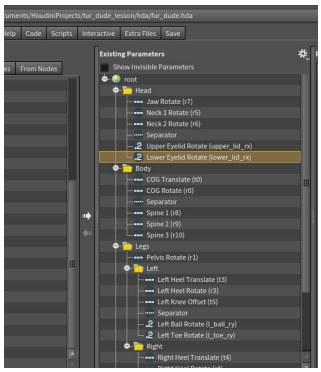
03 Place an **Attach Joint Geometry** node between the *deletejoints* node and the *rigpose* node. Wire the *mergepacked* node into the second input. Set the **Display Flag** on this node and press **Enter** in the scene view. Make sure the **Mode** is set to **Assign Shapes**. Select all the two eyelid joints then **press G** and use your scroll wheel to find the *circle_ctrl* geometry.

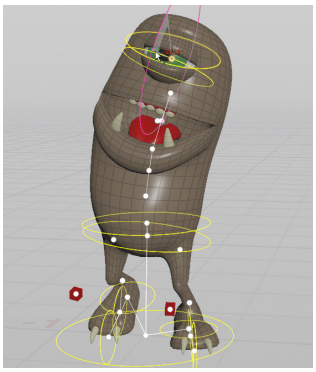


04 In the Operation Control bar at the top, change the **Mode** to **Tweak Shapes**.

Select the *eyelid* joints then **press G** to bring up a transform handle. **Press E** to get the scale handle then click drag on the middle handle to scale in all three directions until these controls are a bit smaller (around 0.5 in the parameter pane).

Now select the *upper_eyelid* joint and move it up by **0.02** then select the *lower_eyelid* joint and move it down by **0.02**.



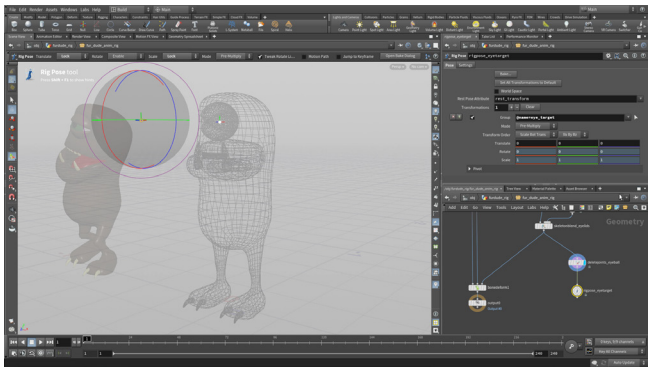


05 From the **Assets** menu, select **Edit Asset Properties > Fur Dude Anim Rig**. Click on the **Parameters** tab.

Lock the **Translate, Rotate** and **Scale** parameters for all of these joints. Unlock the **Rotate X** parameters. From the *rigpose_eyelids* node, drag the *Rotate X* from the *upper_lid* to the **Head** folder. Set the **Range** from **-10** to **30**. Next, drag the *Rotate X* from the *lower_lid* to the **Head** folder. Set the **Range** from **-20** to **20**.

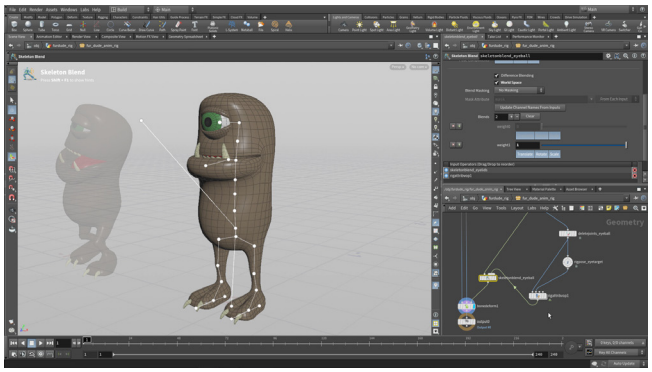
Add a Separator to divide the eye controls from the head controls.

Click **Accept**. This will save the new controls to the rig. You can now explore them using the *test_rig*.



06 Branch off a **Delete joints** node from the *skeletonblend_eyelids* node, set its **Display flag** and name it *deletejoints_eyes*. Click on the arrow beside **Group** and in the scene view select the *eyeball* and *eye_target* joints. Press **Enter** then set **Operation** to **Delete Non-Selected**.

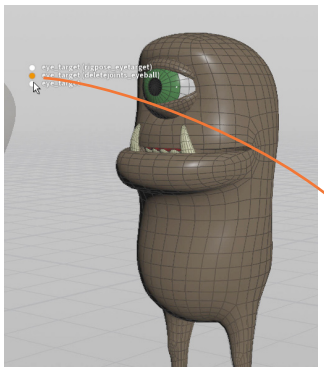
Add a **Rig Pose** node and rename it *rigpose_eyetarget*. Set its **Display Flag** then in the Scene view select the *eye_target* joint. This will add it to the rigpose list. You do not need the *eyeball* joint - it will be controlled by a look at constraint. Lock the **Rotate,** and **Scale** parameters for the *eye_target* joint.

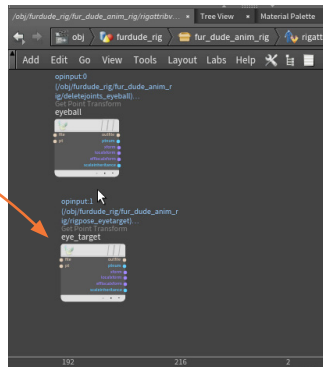


07 Add a **Rig Attribute VOP** node. Wire the *deletejoints_eyes* node into the first input and the *rigpose_eyetarget* into the second one.

Now add a **Skeleton Blend** between the *skeletonblend_eyelids* and *bonedeform* node. Rename the node to *skeletonblend_eyeball* and set the **World Space** checkbox to **On** and *weight1* to **1**. Now feed the output of the *rigattributevop* node into the second input.

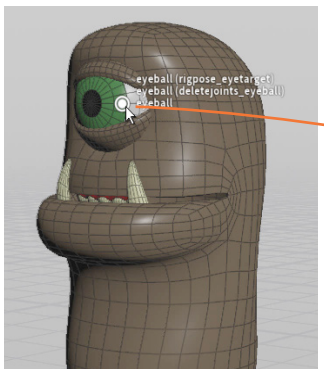
Set the **Display Flag** on the *bonedeform* node.

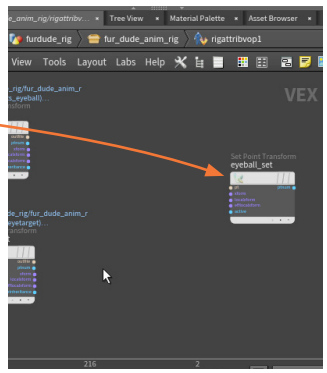




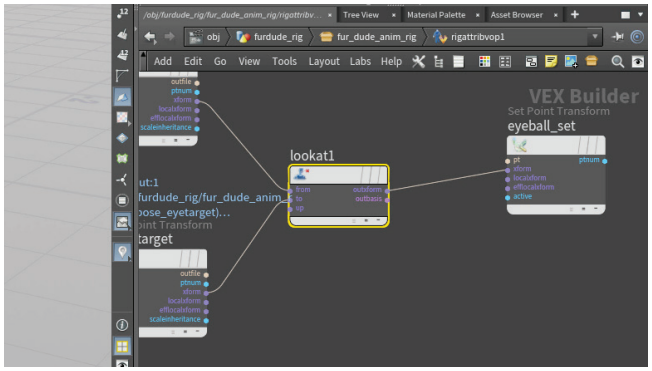
08 Double click on the *rigattributevop* node to dive into it. In the Scene view, click on the *eyeball* joint then drag the *eyeball (deletejoints)* version into the Network editor. This gives you a **Get Point Transform** node that focuses on the *eyeball* joint coming from the **First Input**.

Click on the *eye_target* joint then drag the *eyetarget (ripose_eyetarget)* version into the Network editor. This gives you a **Get Point Transform** node that focuses on the *eye-target* joint coming from the **Second Input.**



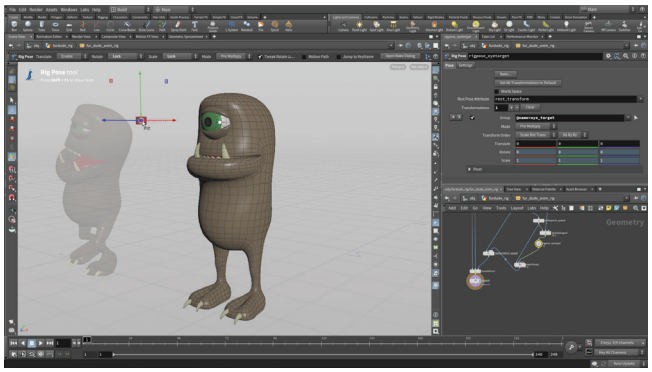


09 Click on the *eyeball* joint then drag the *eyeball* version into the Network editor. This gives you a **Set Point Transform** node that focuses on the *eyeball* joint.



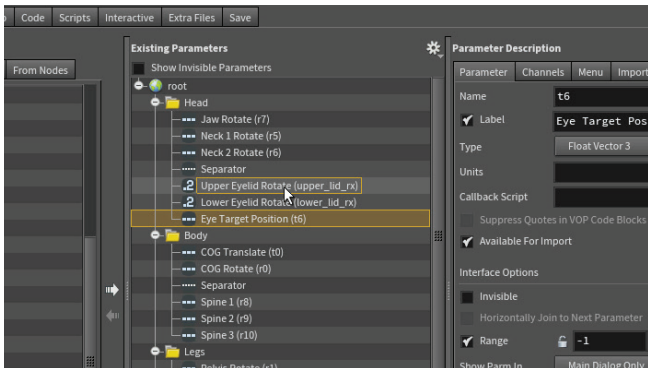
10 Now press **tab > Look At (KineFX)** and place the node in the middle. Wire the **xform** output on the eyeball *getpointtransform* node into the **from** input on the *lookat* node. Wire the **xform** output on the eye_target *getpointtransform* node into the **to** input on the *lookat* node. Wire the **outxform** output on the *lookat* node into the **xform** input on the *eyeball_set* node.

The eyeball geometry flips. Select the *lookat* node and set the **Look At Axis** to **Z** to match the orientation you set on the rig when it was first set up.



11 Put an **Attach Control Geometry** node between the *deletejoints_eyes* node and the *rigpose_eyetarget* node. Wire the *mergepacked* node from earlier in the network into the second input. Set the Display flag on this node and go to the Handle tool. Make sure the **Mode** is set to **Assign Shapes**. Select the *eye_target* joint in the 3D view then **press G** and use your scroll wheel to select the *square_ctrl*.

Set the **Display Flag** on the *bonedeform* node. Click on *rigpose_target*. Select and move the *eye_target* joint. This orients the eyeball. **Undo** to set it back to it's original position.

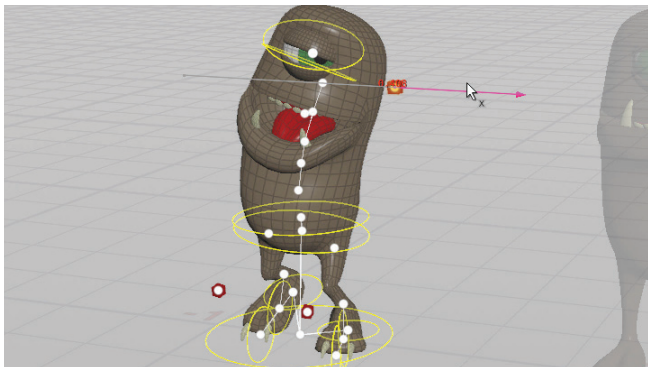


12 Add a **Network Box** around all of the nodes used to define the eyes and name it *Eye Controls*.

From the **Assets** menu, select **Edit Asset Properties > Fur Dude Anim Rig**. Click on the **Parameters** tab.

From the *rigpose_eyetarget* node, drag the *Translate* parameters from the *eye_target* to the **Head** folder in the *eye* section. Name them **Eye Target Position**.

Click **Accept**. This will save the new controls to the rig.



13 You can now explore them using the *test_rig*.

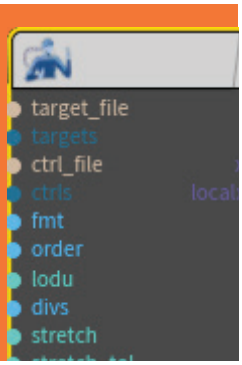
You now have all of the parts finished for this control rig. It is now ready for you to animate a walk cycle. To do this you will create a second copy of the test rig and animate using that network.

This digital asset can create multiple instances in multiple scene files and if you later need to come back and make a change, all of the assets will update. This is the pipeline advantage of working with a digital asset rig.

RIGGING IN VOPS

The Rig Attribute VOP offers a range of different solutions beyond what you have seen in this lesson. IK chains can be built using this approach and the IK chain SOP you used earlier has one of these inside it.

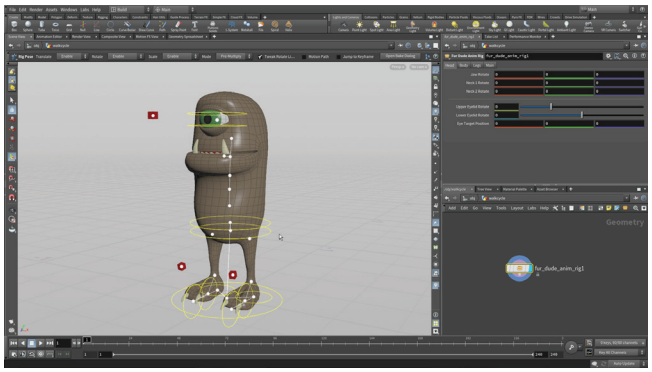
You can also use VOPS to set up a curve solver, realistic shoulder, reverse foot and more. The ability to drag joints from the Scene view to the VOP network is a unique workflow that helps speed up workflow.



PART SIXTEEN

Animate the Rig

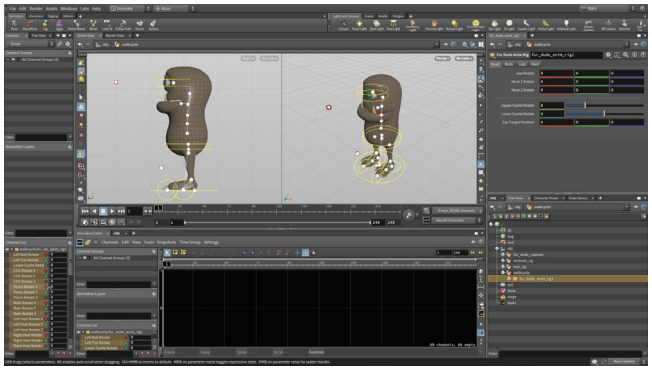
It is time to keyframe a walk cycle for the fur dude character. This will involve new tools such as the Channel List to pin down channels for blocking out the motion. The results will be a quick and dirty walk cycle designed to see fur dude in action. The goal is to map out a basic keyframe workflow to learn a bit about how to animate KineFX rigs.



01 Go to the object level. **Press tab > Geometry** and place down the node. Rename it *walkcycle*. Turn off the **Display Flags** on all of the objects.

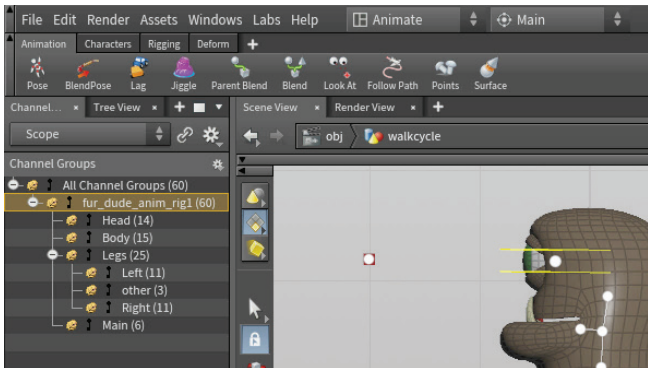
Double-click to dive into *walkcycle* and in the Network view, press **tab > Fur Dude Anim Rig**. Press **Enter** to place it at the origin. This is a new locked version of the fur dude rig that you will animate from scratch.

This puts another version of the character asset into the scene. You can have multiple versions in this scene file or in other scene files and they will all reference the same asset definition on disk.



02 From the **Desktop** menu (the one that currently says Build), choose **Animate**. This gives you panes that are designed to work with a keyframe workflow. You may need to go back into the *walkcycle* object.

The **Channel List** on the left will play a key role in blocking out the animation of the character. The **Animation Editor** lets you display and edit animation curves. In this lesson you will block out the motion and won't be doing any curve editing.



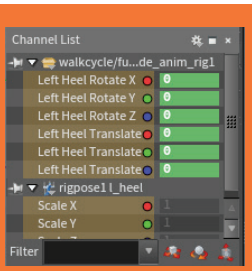
03 In the Network view, select the *fur_dude_anim_rig* inside the *walkcycle* object. Go to the Parameter pane and click on the box icon in the top right. Choose **Parameters and Channels > Create Nested Channel Groups**. Click **Close** in the pop up window. The parameters from your asset are now listed and organized based on their folders.

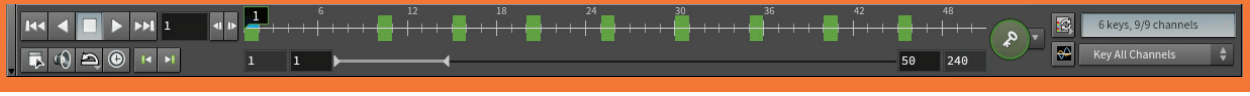
Click on the **Pin** icon next to the *fur_dude_anim_rig* channel group to pin these channels. Make sure your timeline is set to **frame 1** then **press k** to keyframe all the channels.

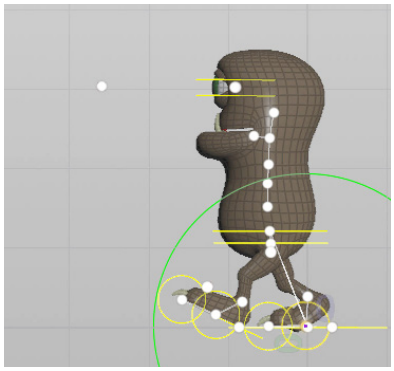
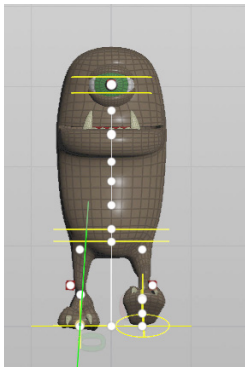
HOW CHANNELS WORK

When you select a joint, the channels load into the channel list. Press **k** to keyframe them.

If you want to keep them loaded in the list for blocking purposes, you can pin them or add them to a channel groups to pin them all together. You can build the Channel groups directly from the asset. They are organized based on how you built the UI for your character. You can also build your own groups to pin down specific channels.

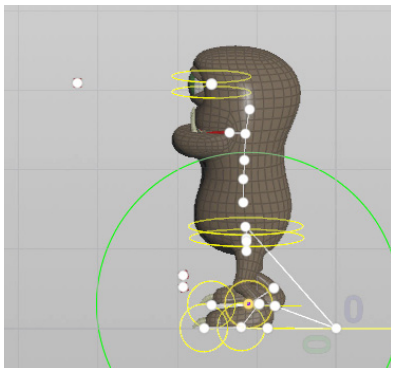
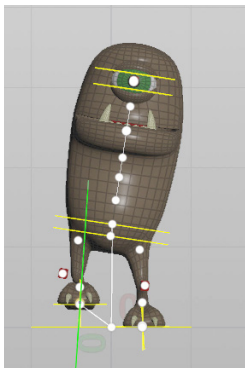




04 Set the **Timeline** to start at **10** and end at **50**. Go to **frame 10**. With all the channels pinned, **press k** to set another keyframe. You want to **keyframe first** then pose. Any posing will update the value for that keyed frame.

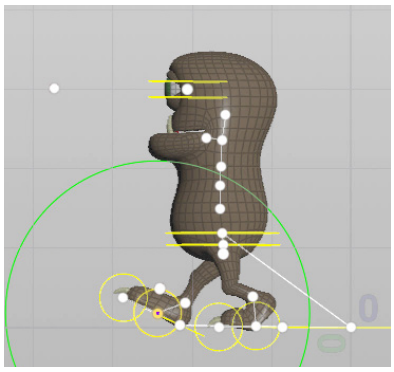
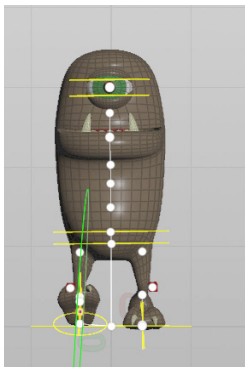
You want to create a pose where the *left heel* moves forward and rotates up. The *COG* will go down a bit and the *right ball* rolls forward.

05 Go to **frame 15** and **press k**.

Roll the *left heel* flat but don't move it. Move the COG to align with the left foot. Move the *right heel* up in line with the other foot. Rotate the *right ball* back to flatten the foot.

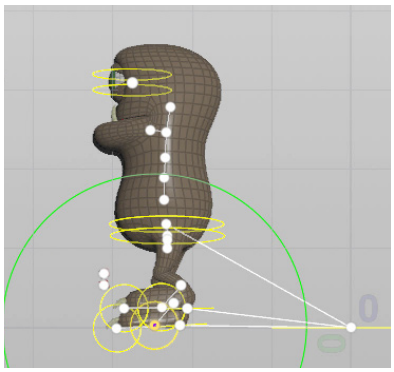
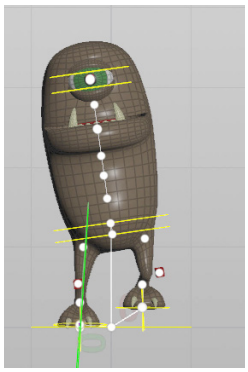
Rotate the *COG* a bit towards the left foot. You can also add some rotation to the three *spine* joints to emphasize this tilt.

06 Go to **frame 20** and **press k**.

Now you want to create a pose where the *right heel* moves forward and rotates up. The *COG* will go down a bit and the *left ball* rolls forward. This is the opposite of the pose at frame 10.

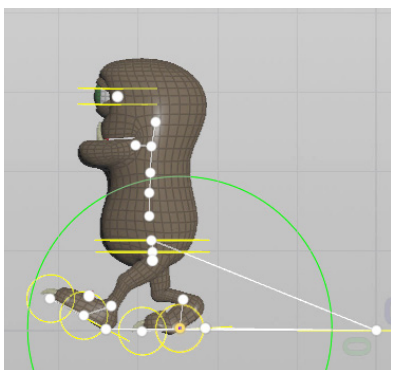
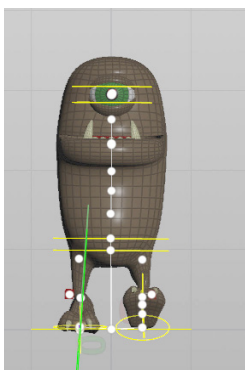
Rotate the *COG* and *spine* joints back to center them.

07 Go to **frame 25** and **press k**.

Roll the *right heel* flat but don't move it. Move the *COG* to align with the right foot. Move the *left heel* up in line with the other foot. Rotate the *left ball* back to flatten the foot.

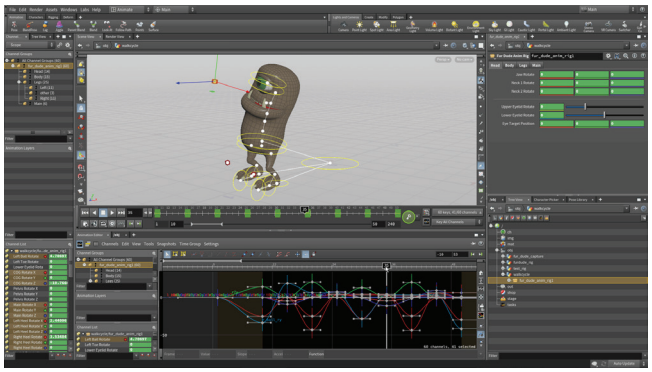
Rotate the *COG* a bit towards the right foot. You can also add some rotation to the three *spine* joints to emphasize this tilt.

08 Go to **frame 30** and **press k**.

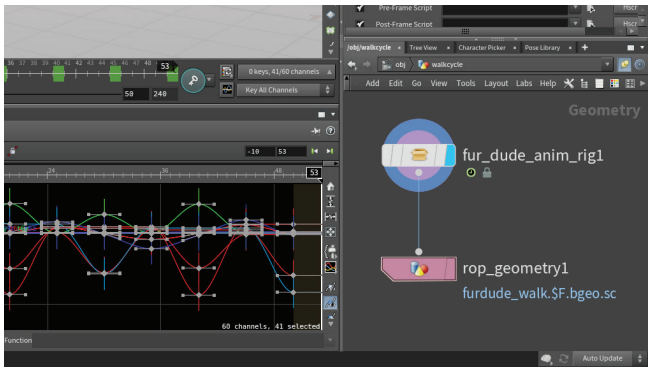
You want to create a pose where the *left heel* moves forward and rotates up. The *COG* will go down a bit and the *right ball* rolls forward.

Rotate the COG and spine joints back to center them.



09 Continue this pattern up until frame **50**. You can repeat the same poses to keep the walk cycle moving forward.

At this point you can go back and tweak any pose to refine the motion. You can also explore creating some overlapping action using extra keyframes. You can animate some eye movement and maybe the eyelids blinking. You can also go beyond 50 frames if you would like a longer animation.

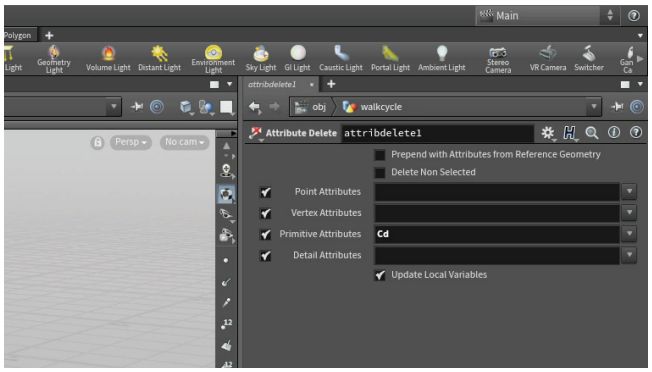


10 Connect the output of the *fur_dude_anim_rig* node into a ROP Geometry node. This will allow you to export out a cache of the fur dude geometry. Set the **Output File** to

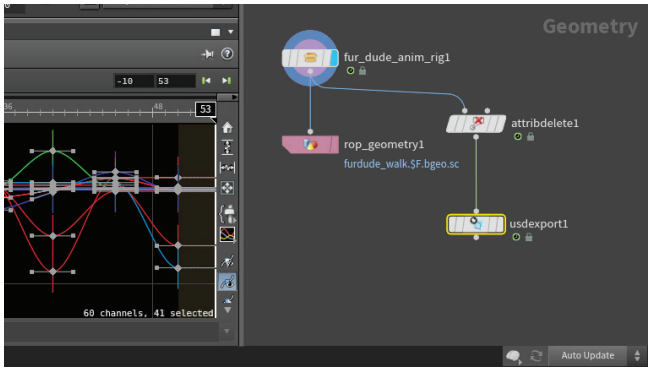
`$HIP/geo/furdude_walk.$F.bgeo.sc`

Next **Set Valid Frame Range** to **Render Frame Range**. **RMB-click** on the **Start/End/Inc** parameter and choose **Delete Channels**. Set the **Start** to **1** and the **End** to **50**.

Click the **Save to Disk** button to store the cache to disk. You will use this to add fur.



11 Branch an **Attribute Delete** node off of the *fur_dude_anim_rig* node. Under **Primitive Attributes** choose **Cd**. This removes the color from all the body parts.



12 Connect the output of the *attributedelete* node into a **USD Export** node. This will allow you to export out a the fur dude geometry to the USD format. Set the **Output File** to

`$HIP/usd/furdude_walk.usd`

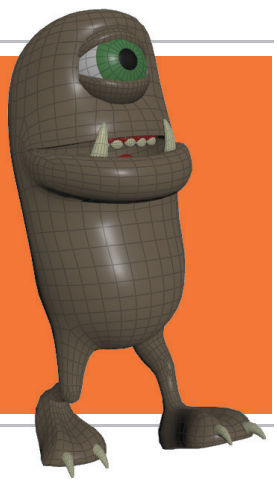
Next **Set Valid Frame Range** to **Render Frame Range**. **RMB-click** on the **Start/End/Inc** parameter and choose **Delete Channels**. Set the **Start** to **1** and the **End** to **50**.

Click the **Save to Disk** button to store the cache out the USD file. You will use this later in the rendering process.

CACHING OUT ANIMATION

Because of Houdini's procedural nature it is not really necessary to cache out the animation. You could reference it from this network to another network for grooming or to Solaris for conversion to USD. The advantage of caching is that you lock down your animation and work with a flattened file on disk. This is a very production-friendly approach.

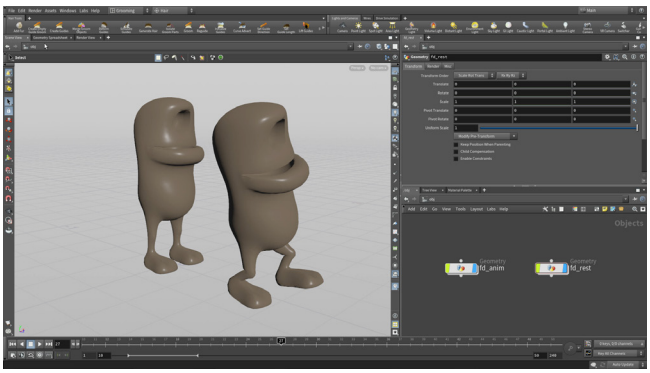
In Solaris a USD file referenced from disk is also more efficient. Since Houdini references files from disk, you are always free to change your animation and output the new sequence and it will be picked up automatically.



PART SEVENTEEN

Add & Groom the Fur

Fur dude gets his name for a reason and you are going to work with a variety of grooming tools to add and shape the hair. Using a desktop designed for grooming, you will add frizz, clumping and hair dynamics which will simulate as fur dude walks. The end result will be ready to be exported for rendering.



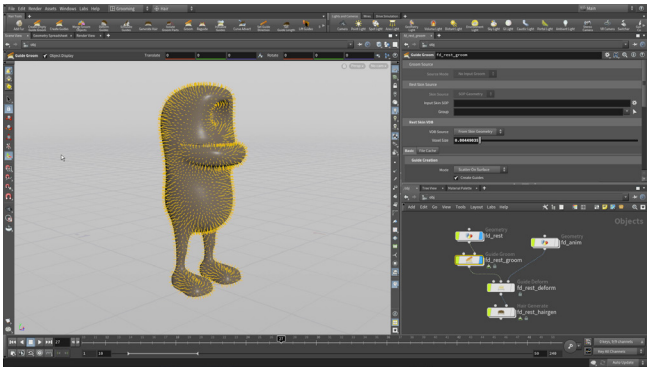
01

Got to the **Grooming** desktop. Put the four existing objects into a network box and call it box *Rig & Animate*.

In the Network editor, press **tab > File**. Place the node then double click to go into it. Click on the **browse** button next to **Geometry File** and go to *$HIP/geo*. Select *furdude_walk.$F.bgeo.sc*. Press **Accept**.

Add a **Blast** node, set it's **Group** to *fur_dude_body* and turn **On** the **Delete Non-Selected** checkbox to focus on the body. Set the **Display Flag** then go to the Object level and rename this *fd_anim*.

Alt drag to create a copy it and name it *fd_rest*. Dive in and change **Geometry File** to *$HIP/geo/ furdude_walk.1.bgeo.sc*.

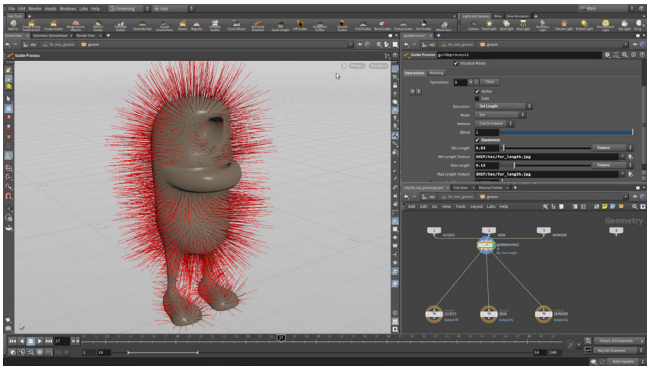


02

Now move the time slide forward a bit. One object has a static version of fur dude and the other is animated. Click the **Add Fur** button . Select the *fd_rest* object and press **Enter**.

Now select the *fd_anim* object and press **Enter**. Turn off the **Display Flag** on *fd_rest_anim*, *fd_rest_deform* and *fd_rest_hairgen*.

Turn on the **Display flag** for *fd_rest* and *fd_rest_groom*.



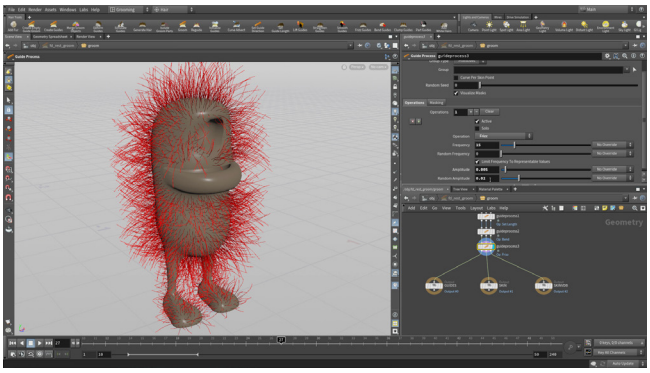
03

Select the *fd_rest_groom* node and from the **Hair Tools** shelf, click on **Set Guide Length**. Turn the Randomize Button to On. Set M**in Length** to **0.03** and from the menu on the right select **Texture**.

Use the file browser button to select *$HIP* then go into the *tex* directory and choose *fur_length.jpg*.

Now set **Max Length** to **0.15** and again choose **Texture**. You can use the **arrow** on the right to select the *fur_length.jpg* image.

You now have the eye, lip and the bottom of the feet masked out and the rest of the fur with random lengths.



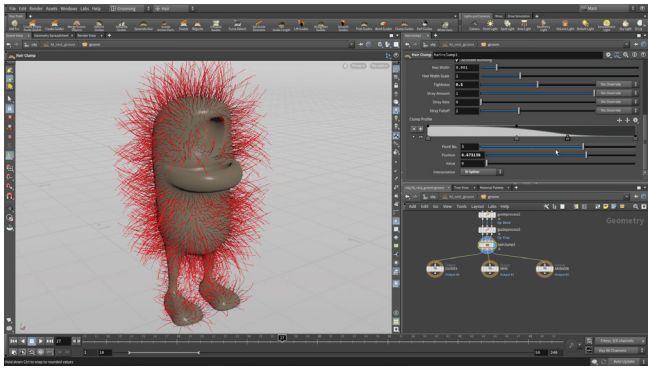
04

From the **Hair Tools** shelf, click on **Bend Guides**. Set **Angle** to **45** to add some bending to the guides.

Next, click on the **Frizz Guides** tool. Set the following:

- **Frequency** to **15**
- **Amplitude** to **0.005**
- **Random Amplitude** to **0.02**

This keeps the hairs from looking too straight when rendered. You can add more frizz if you want a more tangled look.



05

From the **Hair Tools** shelf, click on **Clump Guides**. Set **Clump Size** to **0.02** to **Tightness** to **0.5**.

Next change the **Clump Profile** so that it stays high for a bit longer then tapers off near the end.

Now go to the Object level and turn **off** the display on the *fd_rest* and the *fd_rest_groom* and turn **on** the display on the *fd_anim* and *fd_rest_sim* and *fd_rest_hairgen*.

Select the *fd_rest_groom* node and set **Density** to **20000**.

06

Select the *fd_rest deform* node and from the **Hair Tools** shelf, click on **Simulate Guides**. On the *fd_rest_sim* node, go to the **Vellum Constraints** tab and under **Bend** set **Stiffness** to **5**.

Alt-drag on the *fd_anim* node to make a copy and rename it *fd_collision*. Dive into this node and on the *blast* node set **Delete Non Selected** to **Off**. Add the tongue, upper teeth and gums to the **Group** selection then add a Null node and name it *COLLISION_OUT*.

At the object level, select *fd_rest_sim* then under **Vellum Collisions, set External Collisons to On then** set **Collider SOP** to *../fd_collision/COLLISION_OUT*.

07

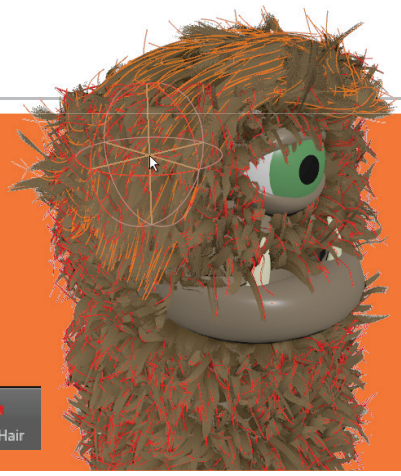
Click on the Caching tab and set **Set Valid Frame Range** to **Save Frame Range**. **RMB-click** on the **Start/End/Inc** parameter and choose **Delete Channels**. Set the **Start** to **1** and the **End** to **50**. Click **Save to Disk** to run the simulation.

Now set the **Load from Disk** checkbox to **On**. The cache will now be used to define the fur instead of the hair calculating for each frame.

08

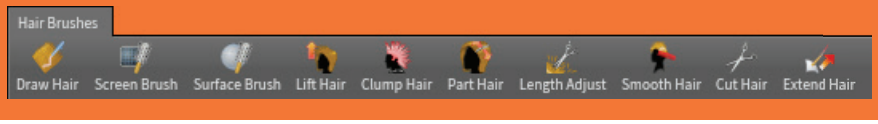
Select the *fd_rest_hairgen* node. Under **Distribution**, set **Density** to **1000000**. Scroll down to Guide Interpolation and set **Clump Crossover** to **0.25** to create a it of overlap between clumps. This gives you an idea of what furdude looks like with a full head of hair.

These are not the hairs that you will render in Houdini's lighting context called Solaris. You will instead bring in the guide hairs and render using a hair procedural at render time.



HAIR BRUSHES

The grooming desktop also has hair brush tools that let you work interactively on the character's surface. You can lengthen, smooth, cut and extend hair. These tools were not needed to set up fur dude's grooming but later you may want to explore them to further style the final look.



PART EIGHTEEN

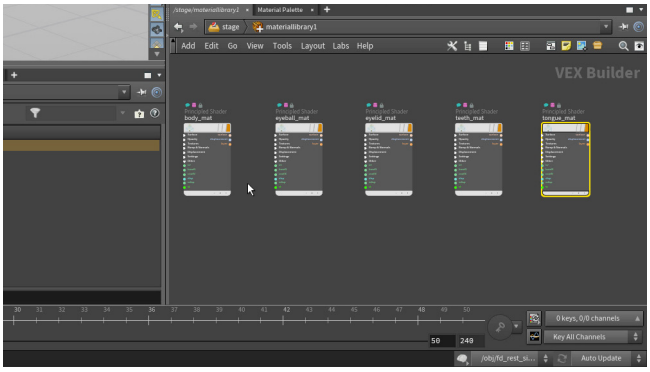
Set up an Render the Shot

To render the shot, you will reference the USD files into the Solaris Stage then add a backdrop. Solaris is a Houdini context that uses LOP nodes to set up a USD Scene Graph. Next, you will import the fur then add and position a camera and a light. The Karma renderer will then be invoked to create a preview render of the shot then render out the animated sequence.

01 Change the desktop to **Solaris**. Choose **Stage** from the path bar. In the Network View press **tab > Reference** then click to add a **Reference** node.

Next to **Reference File**, click on the **File Pattern** and find the *furdude_walk.usd* file. Rename the node to *furdude*. Set the **Primitive Path** to */char/`@sourcename`* - this will use the node name and place it into a group called *char*. In the **Scene Graph Tree** expand *char* then *furdude* to see all of the named primitives.

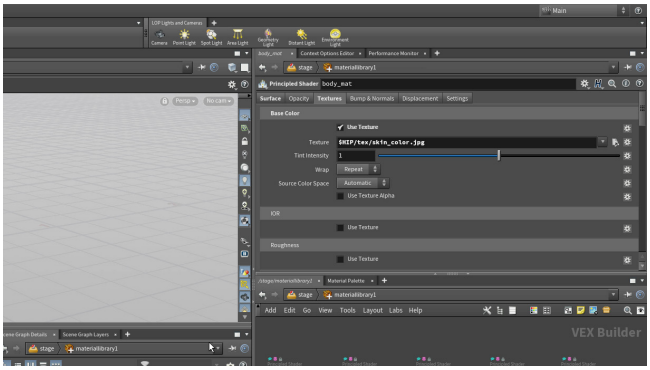
In the Scene View, use your view tools such as **spacebar-h** for homing the view to get a better look at the walk cycle.



02 Press **tab > Material Library. Wire** it into the output of the *reference* node then set its **Display Flag.**

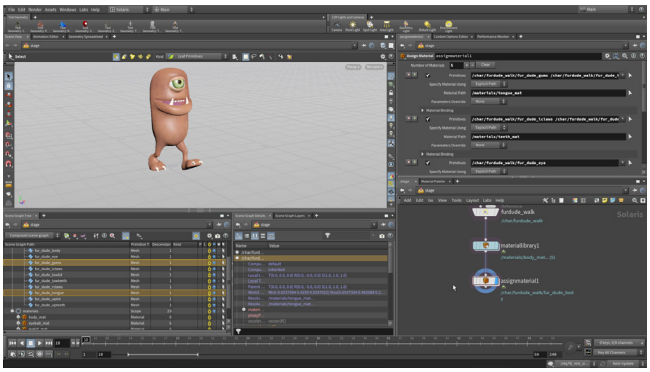
Go to the **Material Palette** pane. Click on the arrow next to */stage/materiallibrary* to open up this area. Scroll through the material gallery on the left of the palette and drag a **Principled Shader** materials into the *materiallibrary* working area.

Go to the Network view and **Alt-drag** this material to create four more. Rename the five materials *body_mat*, *eyeball_mat*, *eyelid_mat*, *teeth_mat*, and *tongue_mat*. You can also see the materials in the **Scene Graph Tree**.



03 For *furdude_body_mat*, under the **Surface** tab, set **Base Color** to **1, 1, 1**. Click on the **Textures** tab and under **Base Color** click on **Use Texture** then use the button next to **Texture** to call up the file window. Click on **$HIP** in the side list then click on the *tex* folder to open it and then click once on *skin_color.jpg* to select it. Click **Accept** to assign the texture to the material. Next set the **Roughness** to **0.5** and **Reflectivity** to **0**.

Assign *eye_color.jpg* and *eye_lid.jpg* to their materials using the same method. Set *tongue_mat* to a **redish pink** and *teeth_mat* to a **yellowish-white**.



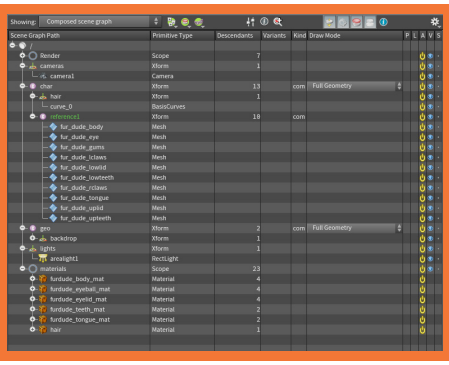
04 Go to the **Stage** level. After the *Material Library* node, add an **Assign Materials** node. From the Scene Graph, drag the *fur_dude_body* to the **Primitives** field then click on the arrow next to **Material Path** and choose *body_mat*. Now click the **Plus Sign** next to add four new entries. Assign them as follows:

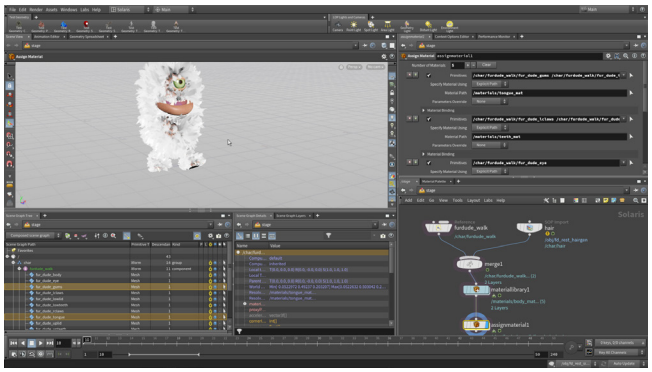
- *fur_dude_eye* > *eyeball_mat*
- *fur_dude_lowid/uplid* > *eyelid_mat*
- *fur_dude_lowteeth/upteeth/claws* > *teeth_mat*
- *fur_dude_tongue/gums* > *tongue_mat*

USD SCENE GRAPH

When working in Solaris, the geometry and materials you add using LOP nodes are added into the **Scene Graph** and converted to USD. When you add a light and a camera, they will also become part of the USD Scene Graph.

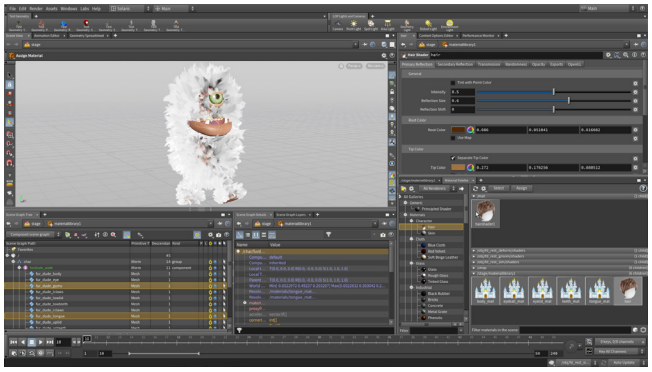
It is not necessary to completely understand USD to light and render in Houdini as an artist but once you start thinking about the pipeline of your project, USD will become a useful tool in managing your shots.





05 In the Network view, press **tab** and type out **SOP Import**. Click to place the node. Rename it *hair*. Set **Import Path Prefix** to */char/$OS*. Click on the node icon net to SOP Path and navigate to the *fd_rest_hairgen* node.

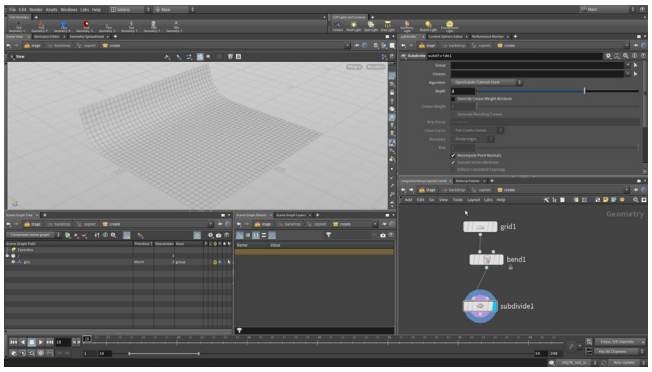
Add a **Merge** node in-between the *furdude* and *materiallibrary* nodes. Wire the *hair* node into it.



06 Go back to the **Material Palette** pane. Open up */ stage/materiallibrary*. Drag the **Hair** material into the *materiallibrary* working area.

Leave the **Root Color** and **Tip Color** set to the defaults for the hair. Next click on the **Secondary Reflection** tab and set **Root Color** to **dark grey** and **Tip Color** to **medium grey**.

Go back to the Assign Material node, From the **Scene Graph**, click on the arrow next to **Primitives** and select the fur curves. Next, click on the arrow next to **Material Path** and choose *hair*.

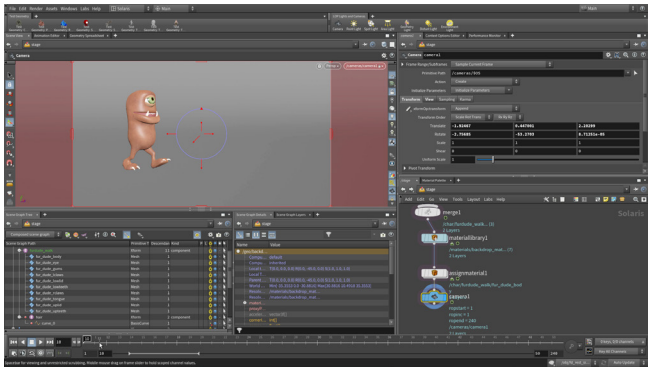


07 In the Network view, press **tab** and type out **Grid**. Click to place the node. Rename it *backdrop* and wire it into *merge*. Set **Import Path Prefix** to */geo/$OS*. **Double-click** on the *backdrop* node to dive down to the geometry level.

Select the *Grid* node and set the **Size** to **50, 50** and **Rows** and **Columns** to **10**. **RMB-click** on the *grid* node's output and type **Bend**. Place the bend node and set its **Display Flag** then set: **Bend** to **75, Capture Origin** to **0, 0, -10, Capture Direction** to **0, 0, -1**, and **Capture Length** to **10. RMB-click** on the *bend* node's output and type **Subdivide.** Set its **Display Flag** then set **Depth** to **2.**

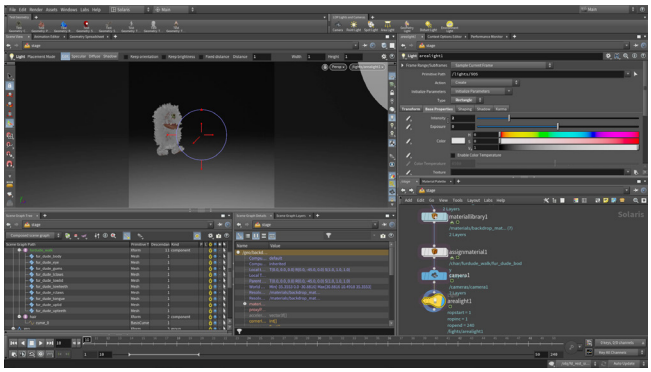
08 Go to object level and set **Rotate Y** to **-45** degrees. Add material and assign it. You can leave the default grey or add your own base color.

Set time range from **10-50**. You will be rendering this sequence after the hair has had 10 frames to settle down.



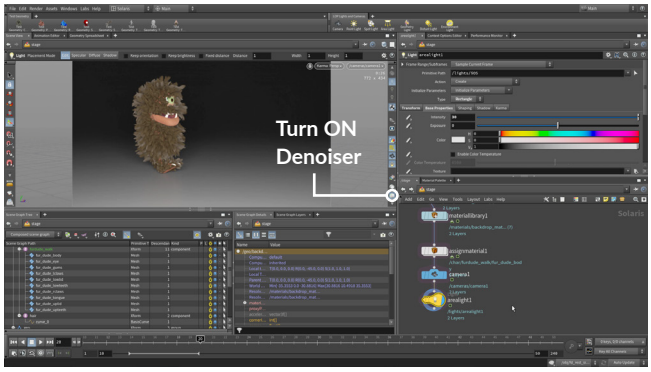
09 Use your view tools to look at *furdude* from the front. From the **LOP Lights and Camera** shelf, **Ctrl-click** on the **Camera** tool. This adds a camera node into the network and you are now looking through the camera in the Scene View.

Press the **Lock Camera/Light to View** button so that view changes can be used to reposition the camera. Now **Tumble**, **Pan** and **Dolly** in the Scene View to reposition the camera so furdude starts on the left and the moves to the right. Scrub the timeline to make sure the camera works for the whole sequence.



10 Now turn **Off** the **Lock Camera/Light to View** button then tumble around until you are looking down on Fur Dude. From the **LOP Lights and Camera** shelf, **Ctrl-click** on the **Area Light** tool. This adds an *arealight* node to the end of the chain.

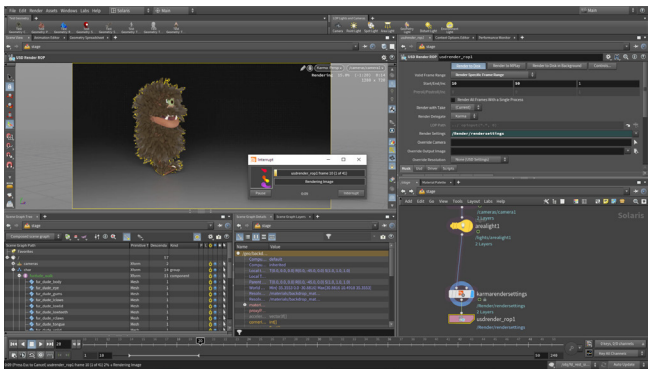
Select the *arealight* node and from the **Base Properties** tab, set the **Intensity** to 2.



11 From the *Persp* menu, choose **Karma** to render with Karma in the Scene View. You can move to different frames in the timeline and the Scene View will update quickly.

Karma is designed to work with USD which is why everything in the LOP context is converted to the USD scene graph. You can only use the Karma renderer from this part of Houdini.

To get a cleaner image when you render, you can turn on the Denoiser if you have an Nvidia graphics card. Be sure to install the **Denoiser** from the **Render** menu and then turn it on in the **Display Options** bar.



12 Press **tab > Karma** to add a **Karma Render Settings** and **USD Render ROP** node. Wire them into the end of the chain. Select *karmarendersettings* and on the **Image Output > Filters** tab set **Denoiser** to **nvidia Optix Denoiser**. Set the **Output Picture** to *$HIP/render/walk/furdude_walk_$F2.exr*. The *$F* in the name is needed to add frame numbers to the renderings and the *2* is the padding of the frame number.

Select *usdrender_rop*. **RMB-click** on the **Start/End/Inc** parameters and choose **Delete Channels**. Set the **Start** to **10** and the **End** to **50** Select the *usdrender_rop* node. Click on **Render to Disk**.

KARMA RENDERER

You are now going to render the sequence using Houdini's renderer Karma. Karma is designed to render USD and is known as a Render Delegate. At first you will render in the Scene View. Press **d** in the Scene View to bring up display options to control your rendering. You can turn on a denoiser, set Pixel Samples and define the Image Resolution.

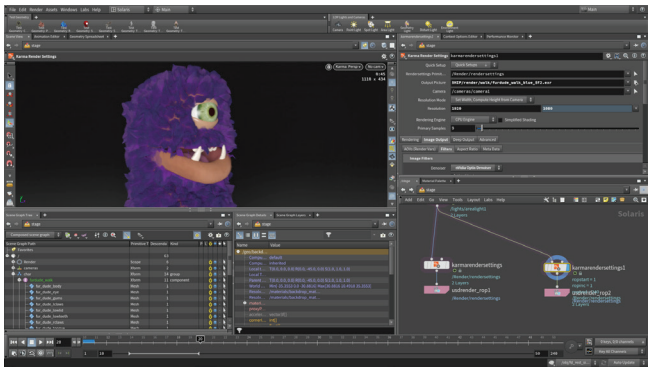
Later when you set up a **Karma LOP**, there will be render settings on that node that you will use to create the final output being saved to disk.

13 When you finish, choose **Render > Mplay > Load Disk Files** and open up the rendered images to review the final sequence.

Later you can branch off another Karma node to up the resolution and render settings for your final rendering. It is always good to complete test renderings at a lower resolution first to make sure that everything is working the way you expect it to.

14 If you want to tweak the hair and fur settings, you can pin the Scene View to the LOP network then go back to the object level and from the **Simulate Guides** node, set the **Load from Disk** checkbox to **Off**. This will allow any changes to flow through to the final rendering.

Here you can see that the hair was shortened and made much frizzier. You can do anything you want. When you are finished be sure to re-cache and then turn **Load from Disk** back to **On**.



15 You can then go back to the Solaris network and maybe change the colors on the fur and re-render.

For the final render you may want to up the **resolution** to **1920x1080** and change some of the quality settings. For instance you can set **Pixel Samples** to **128** and **Light Sampling Quality** to **16**.

CONCLUSION

You have now rigged, animated and rendered the Fur Dude character using the **KineFX** tools in Houdini. Along the way you have touched on various important steps including the creation of a **capture rig** then the layering of a **animation control rig** on top of that. You have packed up the character into a **Houdini Digital Asset** and then keyframed a traditional **walk cycle**.

You then added fur with various grooming tools then rendered using **Karma**. This gives you a complete workflow for creating a shot using this character. You can then go back and refine any aspect of the procedure to create multiple iterations as you seek out the perfect result.

As mentioned before, the KineFX toolset is currently designed for retargeting and motion editing which are not covered in this lesson. These rigging and animation tools will continue to evolve and this lesson provides a taste for what is coming in Houdini's procedural rigging workflow in future versions.

HOUDINI FOUNDATIONS

PROCEDURAL ASSETS FOR UNREAL

To create game assets using Houdini's node-based workflow, it is important to start learning how to think and work procedurally. In this lesson, you will learn how to create game assets using procedural nodes and networks then deploy them directly into Unreal using the Houdini Engine.

Along the way, you will get to use different aspects of Houdini's user interface. You will learn how the different UI elements work together to support you as you build your game assets.

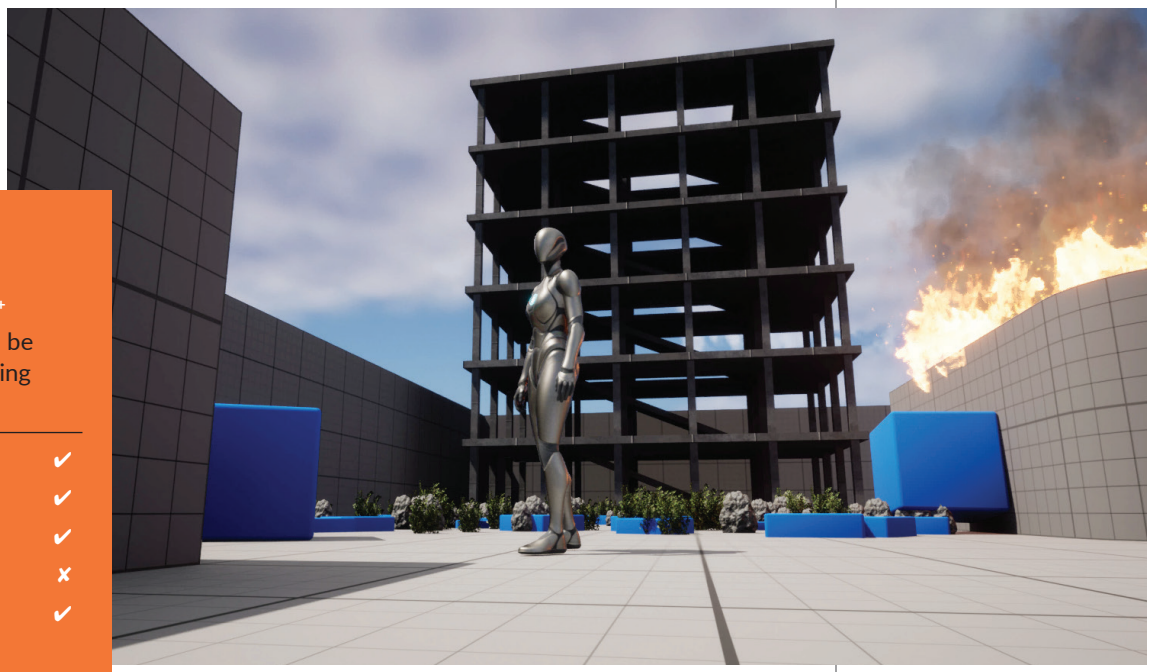
This lesson was completed using **Unreal Engine 5**. Note that while the lesson focuses on Unreal, you can import the same assets into Unity using the Houdini Engine.

LESSON GOAL

To create assets that can be imported into Unreal as game art.

WHAT YOU WILL LEARN

- How to work with **Nodes and Networks** to control the flow of data
- How **Houdini Digital Assets** can be used to package and share your solution with others
- How to load Houdini Digital Assets into the **Unreal Editor**
- How to **instance** objects to points and control the orientation and scale of the objects using attributes
- How to create a game asset with **Collision Geometry** for use in Unreal
- How to export a **Rigid Body Simulation** as an FBX file for Unreal.



LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

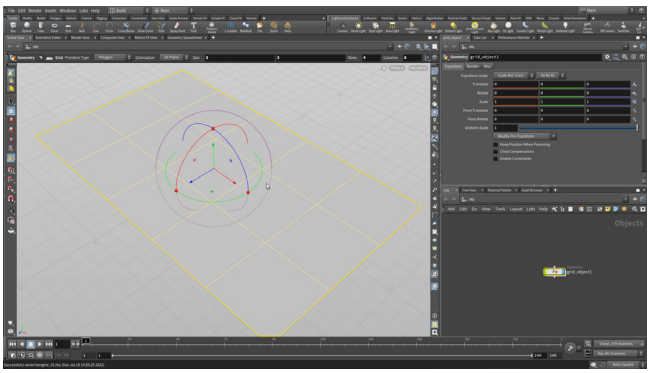
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✘
Houdini Education	✔

Document Version 3.0 | July 2022

PART ONE

Create a Simple Building

Learn how to build a simple building using a procedural network of nodes. Some of the nodes will be created by interacting in the Scene View and others in the Network view. You will use channel references to connect parts of one node with other nodes in the system. This creates a procedural solution that you will wrap up into a Houdini Digital Asset.

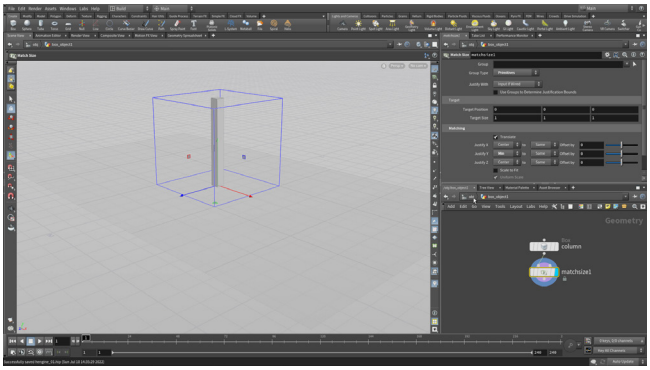


01 Select **File > New Project**. Call it *hengine_lesson* and press **Accept**. Select **File > Save As...** Set the file name to *hengine_01.hip* and click **Accept** to save.

In the viewport, **press c** and choose **Create > Geometry > Grid**. **Press Enter** to place it the grid at the origin. In the **Operation Controls** bar at the top of the Scene View, set:

- **Size** to **5, 3**
- **Rows** to **4**
- **Columns** to **6**

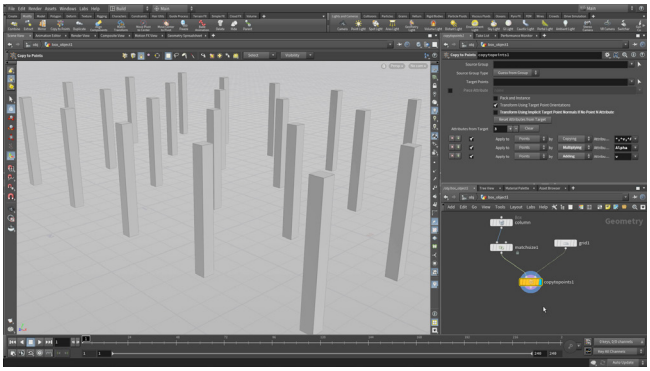
Press v and choose **Shading > Smooth Wire Shaded.**



02 **Press c** and choose **Create > Geometry > Box**. Press **Enter** to place it at the origin. **Double click** on the *box_object* in the Network view. This puts you inside the object at the geometry level. Set the following:

- **Size** to **0.1, 1, 0.1**

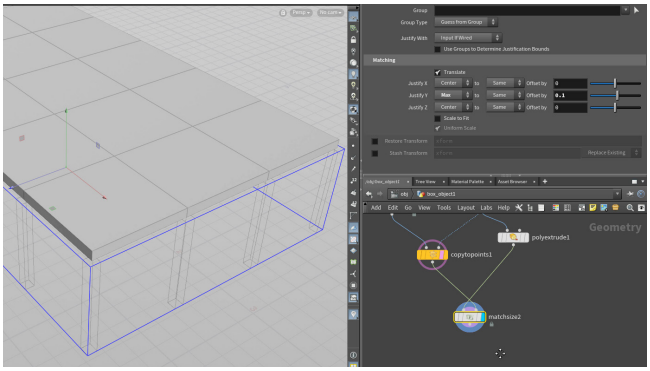
Rename the *box* node to *column*. In the Scene view press tab and start typing **Match Size** then select **Match Size** . **Press n** to select the box and then press **Enter**. This adds a new node. In the Parameter pane, under **Matching**, set **Justify Y** to **Min**. This raises the box up so that it sits on the ground.



03 **Press u** to go back up to the object level or click on **obj** on one of the Network Path bars. Click on the **Select** tool and click in empty space to deselect all the objects.

On the **Modify** shelf, click on **Copy to Points.** Select the column as the *geometry to copy* and press **Enter** then click on the grid as the *geometry on whose points to copy* and press **Enter.**

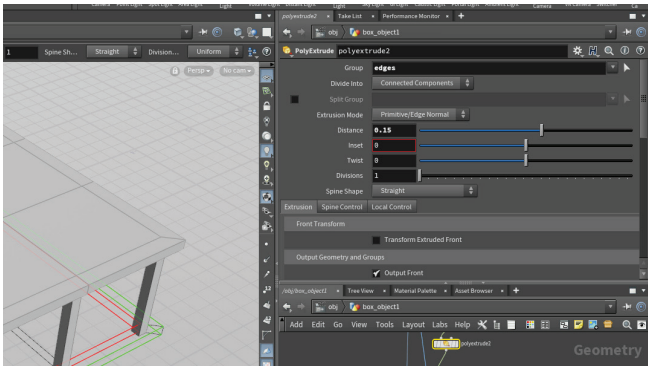
In the Parameter pane, set **Transform Using Implicit Target...** to **Off**. Now the columns will be pointing up. Turn on **Pack and Instance** to ensure instanced geometry in Unreal. **Press 4** to go to primitive selection mode. This will hide all the corner points.



04 Press **tab > PolyExtrude** in the Network View. Place it to the side. Wire the grid node into the *polyextrude* node and set its **Display flag**. Set **Distance** to **0.1** and under **Output Geometry and Groups** turn on **Output Back**. Set the **Template Flag** on the **copytopoints** node.

Add a **Match Size** node to the network and wire *polyextrude* into the first input and *copytopoints* into the second. Set **Justify Y to Max to Same**, **Offset** by **0.1.** Now the extruded shape will sit on top of the columns.

Add a **Merge** node and wire *copytopoints* and *matchsize* into it.

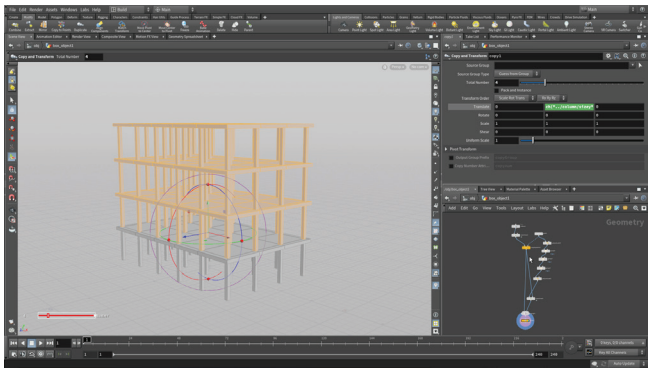


05

In the Network view, press **tab > group**. Wire it between *polyextrude* and *matchsize*. Change **Group Name** to *edges*.

Alt drag to create a another *group* node and set its **Display flag.** Keep **Group Name** set to *edges* and then set **Initial Merge** to **Subtract from Existing**. Turn off **Enable** under **Base Group** and turn on **Enable** under **Keep by Normals**. Set **Direction** to **0, 1, 0** and **Spread Angle** to **0**. **Alt drag** on this node to make a copy and wire it into the chain. Change **Direction** to **0, -1, 0**. Now only the sides of the box are in the *edges* group.

Add a **PolyExtrude** node. Set **Group** to *edges* and **Direction** to **0.15**.

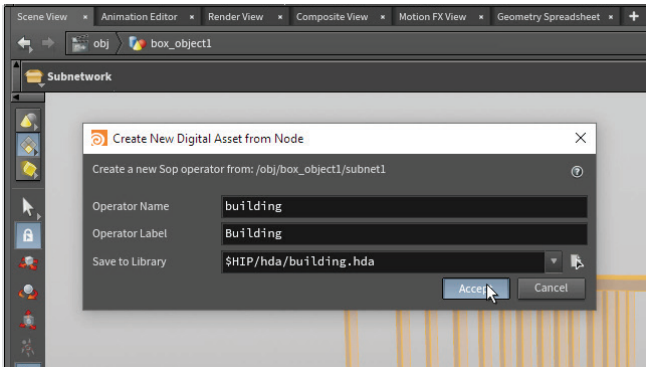


06

Set the *merge* node's **Display Flag** to see the first floor of the building complete. Turn off the **Template** Flag on the *copytopoints* node.

In the Scene view, **press n** to select all and then press **tab > Copy and Transform**. Select the *column* node and **RMB-click** on **Center Y** and choose **Copy Parameter**. Go to the *copy* node and **RMB-click** on **Translate Y** and choose **Paste Relative References**. Add a **+ 0.1** to the expression.

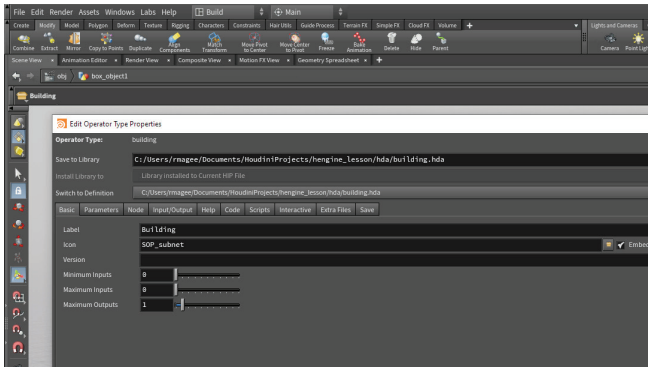
Now increase **Total Number** to 4 to add three more floors.



07

Select all the nodes in the Network editor. From the **Asset** menu, choose **New Digital Asset from Selection**. This will collapse the network into a subnetwork then use that subnet node to create the Digital Asset.

Set the **Operator Name** to *building* which will change the **Operator Label** to *Building*. Click on the button to the far right of **Save to Library.** In the *Locations* sidebar, click on $HIP/ and then double-click on the *hda* directory. Press **Accept** and then **Accept** again to save the asset to disk.



08

The **Edit Type Properties** window opens up. This panel is where you will build the user interface for your asset. You will revisit this window later. Click **Accept** to close this window.

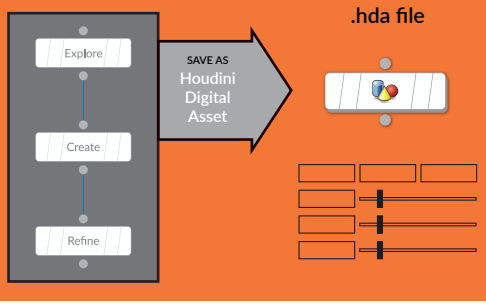
Rename the node in the network view to *building*.

The nodes you have used to build the asset will remain a part of the asset even after you save it. This will allow you to continue making changes even after you have started using it in your Unreal game levels.

WHAT IS AN HDA FILE?

Houdini nodes and networks can be encapsulated into single nodes called **Houdini Digital Assets** which let you share your techniques with colleagues. These assets are saved to disk inside files known as **.hda** files.

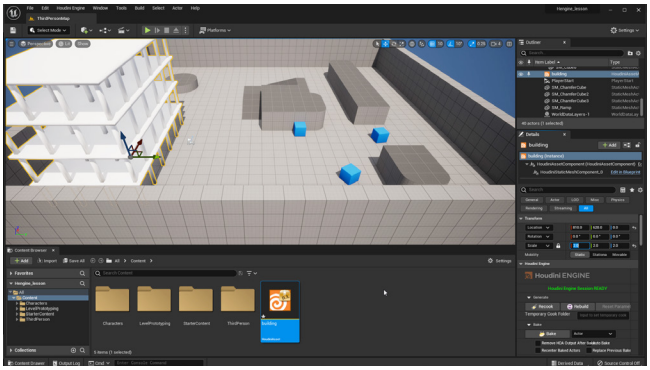
Asset files created in older versions of Houdini may have a different extension **.otl** which means Operator Type Library - both these file types work the same way.



PART TWO

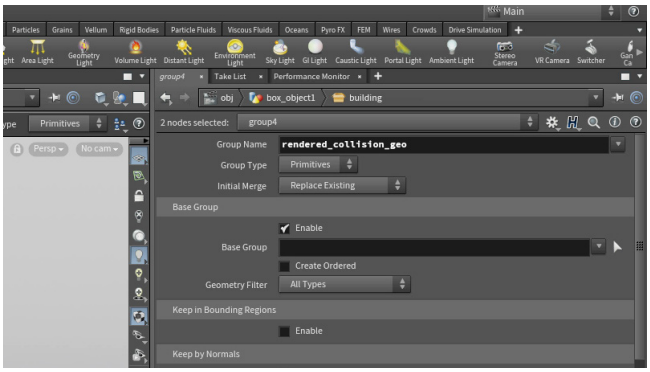
Import the Asset into Unreal

Now that you have a digital asset file on disk, you can import it into the Unreal Engine. This is possible because of the Houdini Engine plug-in which connects the two applications. Houdini Digital Assets that are loaded into Unreal are cooked using Houdini under the surface. **The Houdini Engine for Unreal plug-in must be installed to complete this lesson (See Sidefx.com/unreal for instructions).** *Please note that Houdini Apprentice does not work with Engine.*



01 **IN UNREAL** – Set up a **Third Person Template** with **Blueprint** support. Delete the *TextRenderActor*. Open the **Content Drawer** and click on the **Import** button. You may want to dock the **Content Drawer.** Navigate to your Houdini project and find the *building* asset. Click **Open**.

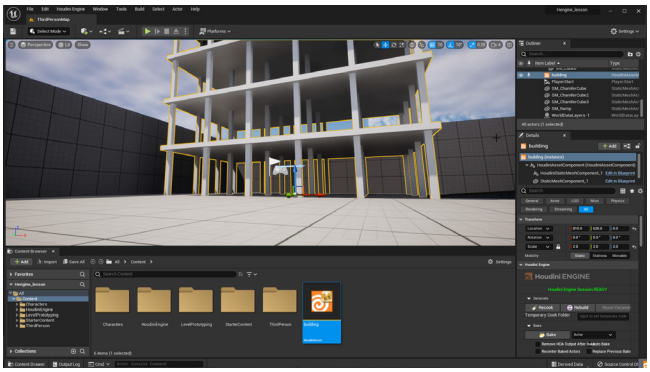
Set the **Scale** to **2, 2, 2** and move it into the corner. If you press the **Play** button you will see the building during gameplay.



02 **IN HOUDINI** – To add normals and collisions to your geometry you need to add some nodes to the network.

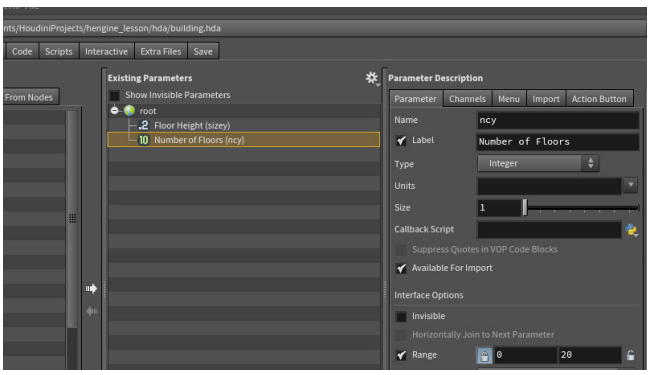
Dive into the *building* network and between the *polyextrude* and *matchsize* nodes, add a **Normal** node. This will add normals for the geometry to display properly in Unreal. Under the *normal* node add a **Group** node and set the **Group Name** to *rendered_collision_geo*. This will turn your geometry into collision geometry.

Copy and Paste these two nodes and insert the new nodes after the *column* node.

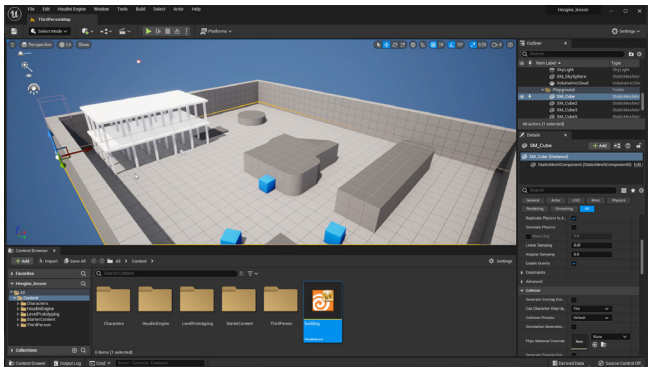


03 **IN UNREAL** – In the Generate section of the *building* asset's **Details** panel, press **Rebuild**. Press **Play** to explore the scene.

The normals are working properly and now you will collide with the columns if you try to run through them.



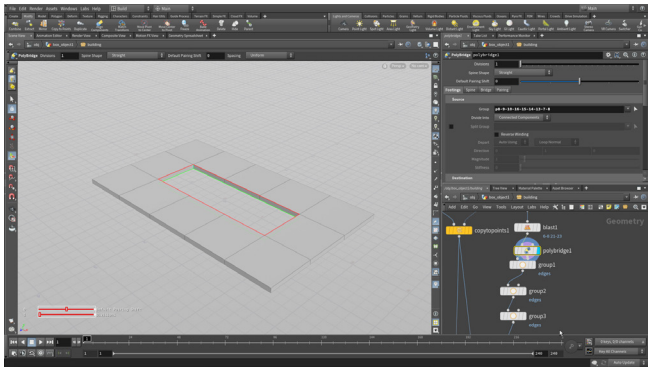
04 **IN HOUDINI** – Select **Assets > Edit Asset Properties > Building**. In the properties window, click on the Parameters tab. Now select the *column* node and drag the **Size Y** parameter to the Parameter tab. **Rename** it **Floor Height**. Now from the *copy* node, drag the **Total Number** parameter over and rename it **Number of Floors**. Click **Accept**.



05 **IN UNREAL** – Press **Rebuild** in the building asset's **Details** panel. Scroll down to see that there are now parameters for **Floor Height** and **Number of Floors**.

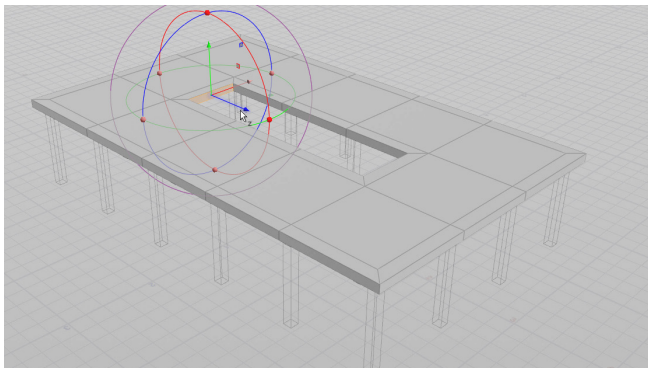
Set **Floor Height** to **1.5** and **Number of Floors** to **2**.

You can **Play** the level to walk around the asset and review the changes.



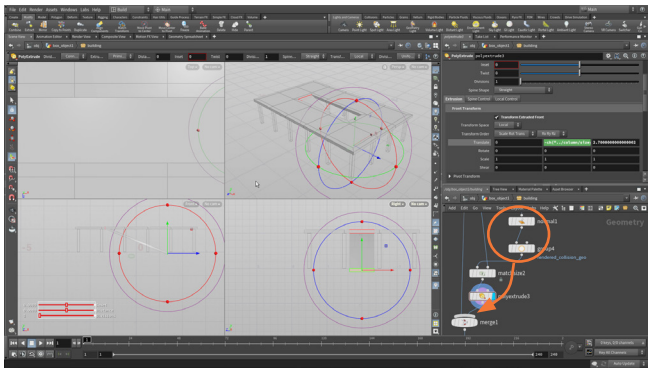
06 **IN HOUDINI** – Go back and set the **Display flag** on the floor slab's *polyextrude* node. **Press 4** to get primitive selection. Select the middle three primitives from the top then tumble around and shift-select the three primitives from the bottom of the slab. Press **Delete**. This adds a **Blast** node.

Press 3 to get edge selection. **Double click** on the bottom edge of the hole to select the whole loop. Go **tab > Polybridge.** Press **Enter**. Now **double click** on the top edge of the hole then press **Enter**. This will add geometry to the inside of the hole.



07 Set the **Display Flag** on the *matchsize* node. You will see the new hole with an overhang.

Press 4 to get primitive selection. Get the **Select** tool then select the end face of the opening. It may not appear selected but it is. Press **tab > PolyExtrude**. Under Extrusion, turn on **Transform Extruded Front.** Start pulling the face out and down.

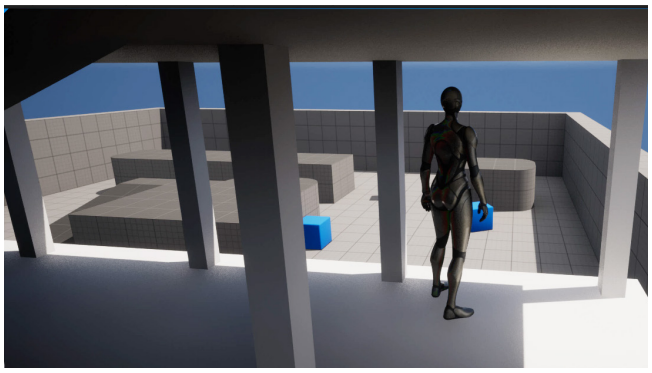


08 Copy the *column* node's **Size Y** channel and **Paste Relative Reference** to the new *polyextrude* node's **Translate Y**. Add a - sign to the expression then subtract 0.1. The expression should read: `-ch("../column/sizey")-0.1`

Set **Translate Z** to **2.7**.

Move the *normal* and *group* nodes to after this *polyextrude* node. Otherwise, the new extrusion won't be collision geometry and the ramp won't work properly.

Set the **Display Flag** on the *output* node. Choose **Assets > Save Asset > Building** to save the changes to the HDA file on disk.

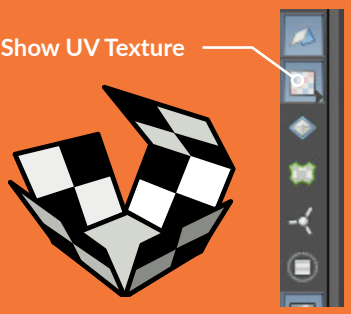


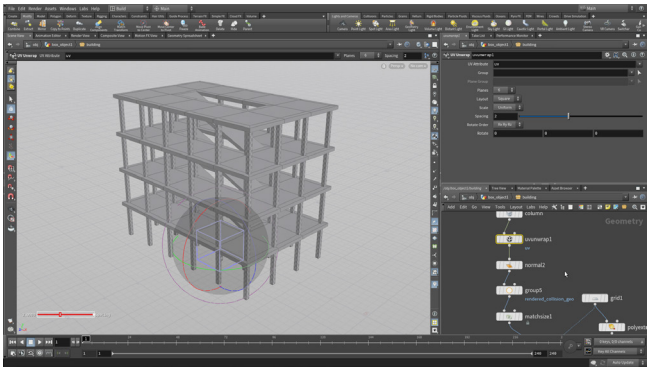
09 **IN UNREAL** – Press **Rebuild** then set the **Floor Height** to **1** and the **Number of Floors** to **6**. Press **Play** and walk up ramps and around the building.

UVS

Houdini has a number of nodes at the geometry level for setting up and managing UVs. In this lesson, you will use **UV Unwrap** and **UV Transform** to add UVs to the building. These work well with the simple shapes used to model the building. For more complex shapes you will use tools such as **UV Flatten** and **UV Layout**.

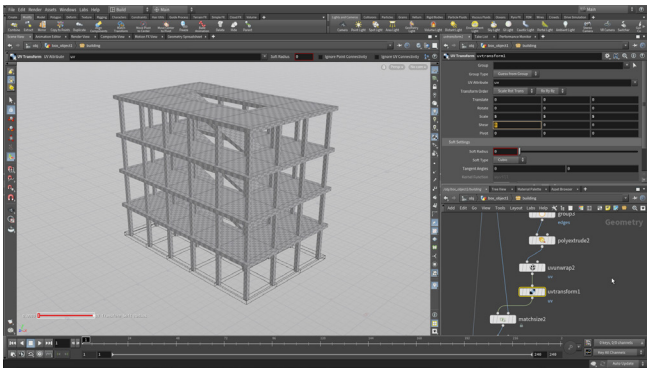
You can see a UV Grid on your geometry once UVs are in place. To hide the grid, you can click on the **Show UV Texture** button on the **Display Options** bar to toggle it on and off.





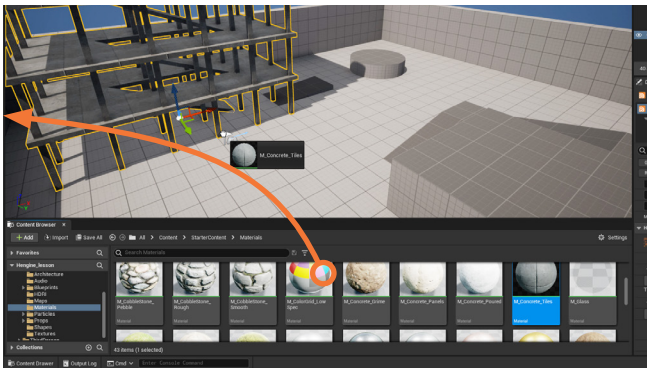
10 **IN HOUDINI** – You need to add some UVs to the geometry for use in the game editor. Go to the part of the network with the *column* node. Press tab> **UV Unwrap** and place the node between the *column* node and the *matchsize* node.

Go to the part of the network with the *second polyextrude* node. Press **tab > UV Unwrap** and place the node between the *polyextrude* node and the *matchsize* node.



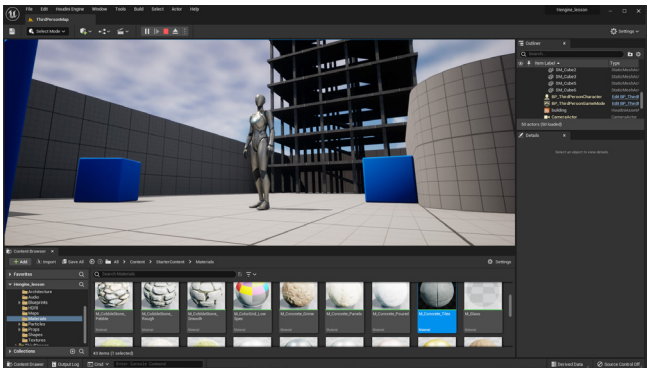
11 You will see a UV grid on the columns. They are a different scale compared to the columns. Press **tab > UV Transform** and place the node between the *uvunwrap* node and the *matchsize* node. Set the **Scale** values to **5, 5, 5**. Now the UVs appear more alike.

Choose **Assets > Save Asset > Building** to save the changes to the HDA file on disk.



12 **IN UNREAL** – Press **Rebuild.** You won't see the UVs until you add a Material.

In the **Content Browser,** navigate up to **Content > Starter Content > Materials**. Make sure the *Building* asset is selected then add a Material such as *M_Concrete_Tiles* to the **building geometry** in the scene view. You may need to drag it to both the columns and the slabs.

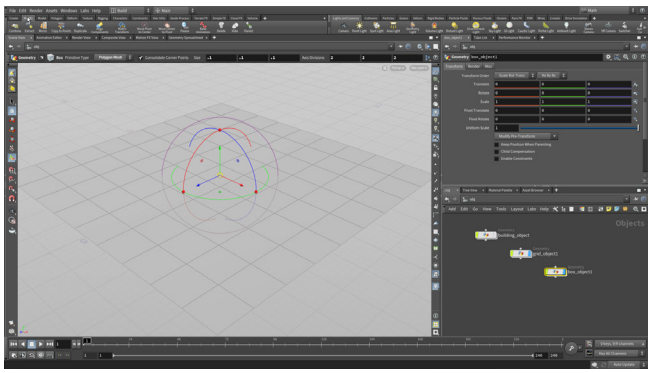


13 Press **Play** to walk through the level and explore the textured building.

This asset could be used more than once on a level to create different buildings with different floor heights and building heights. You could also promote more parameters to the asset to add more control. All the assets on your level that come from this digital asset can all be updated at once when a change is made to the asset's network or its parameter interface.

PART THREE
Copy to Points

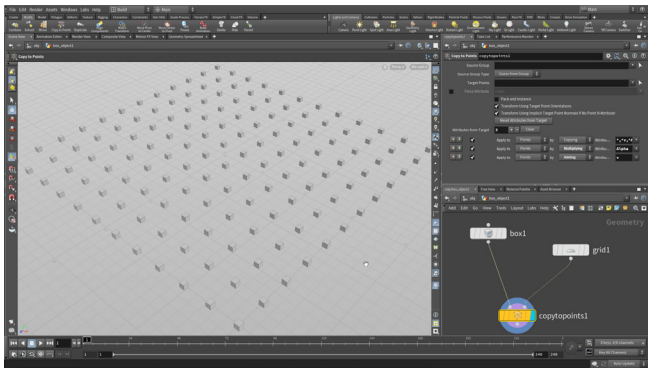
In this part of the lesson, you are going to copy some boxes to points of another grid. You will then randomize those points to achieve a more organic look and add random attributes to rotate and scale the boxes to create more variety in the distribution of the shapes. This creates another procedural system that you will wrap up into a Houdini Digital Asset.



01

IN HOUDINI – Go to the Object level and hide the building. In the viewport, press the **c** key to bring up a radial menu and select **Create > Geometry > Grid**. Release and then press **Enter** to place at the origin. You will start by using the points on this grid surface to instance geometry.

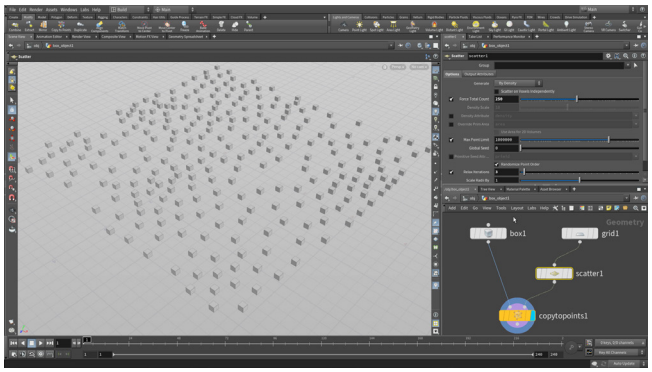
Use the same radial menu to **Create > Geometry > Box** and again press **Enter** to place at the origin. In the **Operation Control** bar at the top of the viewport, set the **Size** to **0.1 0.1 0.1**. This is the geometry that you will be copying to the points on the grid.



02

With the *box* still selected, go to the **Modify** shelf and click on the **Copy to Points** tool. Select the *grid* and press **Enter**. Now the boxes have been copied to all of the grid points.

You can now adjust the look of the system by editing parameters. Select the *grid* node and change the **Size** to **6, 6** and the **Rows** and **Columns** to **12**. The grid gets bigger and has more points but the boxes aren't being copied to all the points. Click on the *copytopoints* node where you can see that **Target Points** are set to **0-99** to match the points on the original grid. Remove the **0-99** to copy boxes to the whole grid.



03

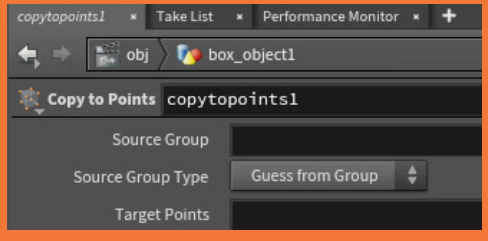
In the Network Editor, press **tab > Scatter**. Place the *scatter* node in the Network editor between the *grid* and the *copytopoints* nodes. It will wire itself into the network automatically and the boxes will now be copied to these new points.

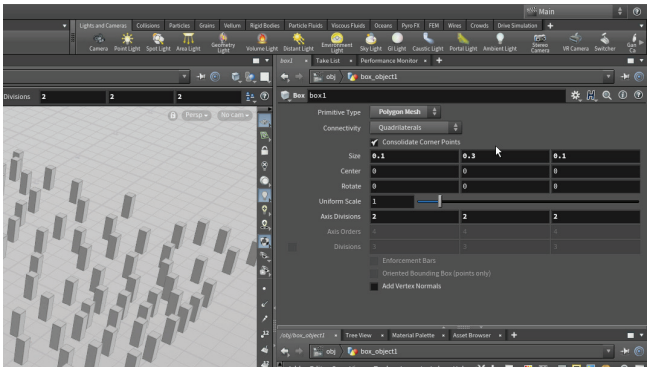
Set **Force Total Count** to **250**. Play with the **Relax Iterations** to adjust how the points are being laid out. The scatter node gives you a more organic layout compared to the original grid points.

SOURCE GROUP

When you model in the Scene view, your selected geometry will get placed into the **Source Group** field as either numbered points or primitives depending on the tool. If the field is empty then the tool will work on **ALL** of the points or primitives.

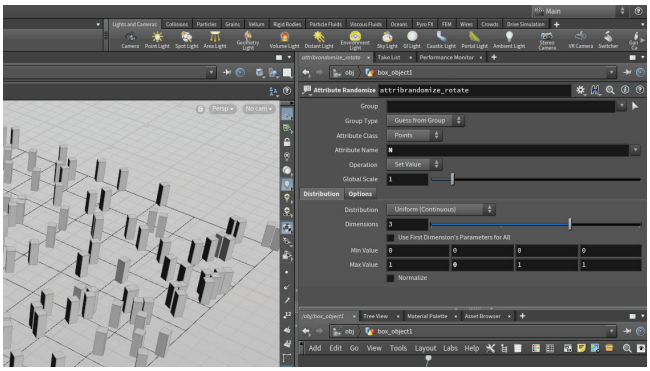
When you use tools at the geometry level this field will be left empty when you choose **Select All [n]** before using the tools. Because you set up *copytopoints* at the object level, it filled in the points as 0-99.





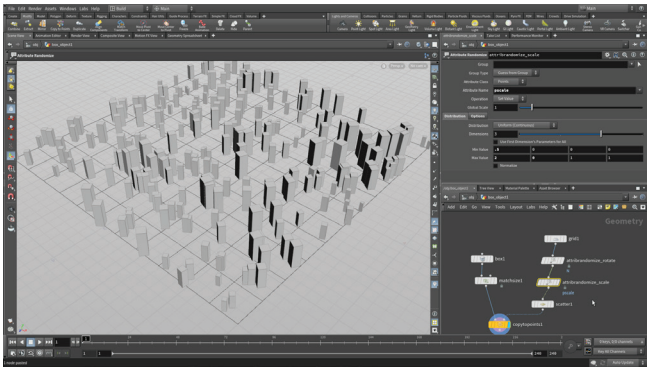
04 In the Network view press tab and start typing **Match Size** then select **Match Size** . Place this node between the *box* node and the *copytopoints* node. In the Parameter pane, under **Matching**, set **Justify Y** to **Min**. .

On the *box* node, set **Size Y** to **0.3** to test the expression. You can see that all of the boxes are sitting on the ground.



05 In the Network Editor, press tab and begin to type **rand...** then choose the **Attribute Randomize** tool. Place the *attributerandomize* node between the *grid* and the *scatter* nodes. At first, you will see random colors on your boxes because the default attribute is color (Cd).

Set the **Attribute Name** to **N**. This changes the attribute to the normal direction and all the boxes are now pointed in different directions. Set **Max Value Y** to **0** which will limit the randomness to the X and Z directions. Rename it *attrrandomize_rotate*.

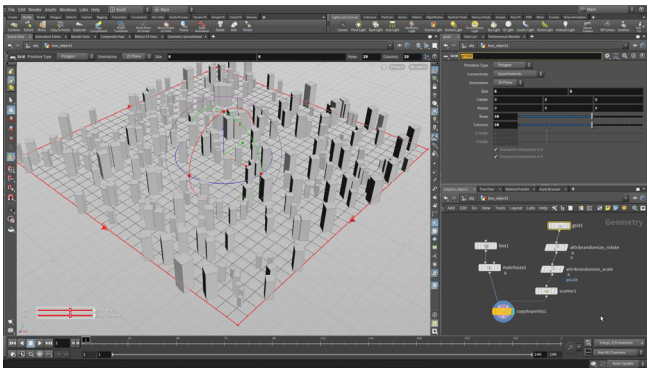


06 Press the **Alt** key and drag on the *attrrandomize_rotate* node to make a copy of it. Drop it between the *attrrandomize_rotate* and the *scatter* nodes. Rename it *attrrandomize_scale*.

Set the following:

- **Attribute Name** to *pscale*.
- **Min Value** to **0.5**
- **Max Value** to **2**.

This gives you a nice variety of sizes for the boxes on the grid.



07 Select the *grid* node and increase the **Rows** and **Columns** to **30** – this creates more points on the grid which creates a wider variety of values on the scatter points.

Save your work.

HOW ATTRIBUTES WORK

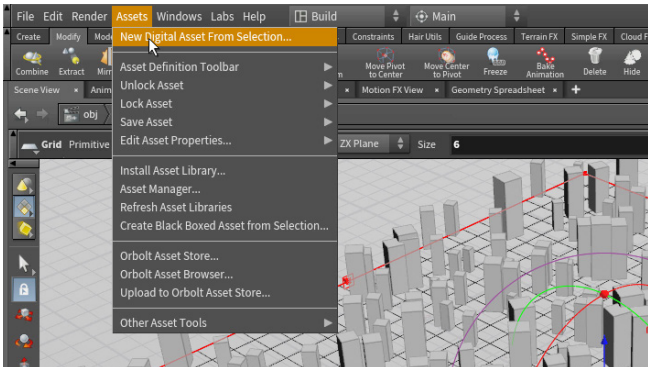
Attributes that are assigned to geometry are passed down the network chain to different nodes. In the case of this network, the attributes being assigned to the grid geometry are passed on to the scattered points which in turn affects the copied geometry. This is an important way to control the flow of data in Houdini.

The attributes are initially assigned to the points on the grid therefore more points lead to more randomization in the attribute values.

PART FOUR

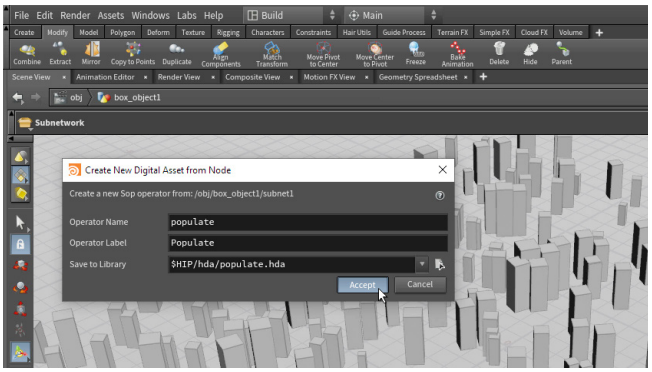
Create another Houdini Digital Asset

In this part of the lesson, you are going to create a digital asset and test the system in Unreal. Just like the building this means wrapping up the network and saving the results to disk as an HDA file. You will promote some parameters to allow you to control the grid size, number of points and the relaxation. The parameters will then be available to you in Unreal.



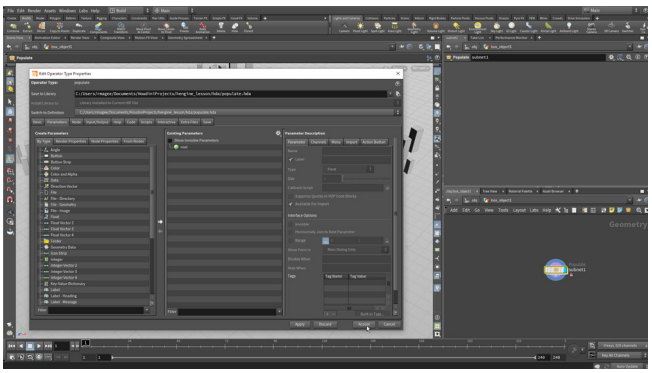
01 Select all the nodes in the Network editor. From the **Asset** menu, choose **New Digital Asset from Selection**. This will collapse the network into a subnetwork then use that subnet node to create the Digital Asset.

The nodes you used to build the asset will remain a part of the asset even after you save it. This will allow you to continue making changes even after you have started using it in your game levels.



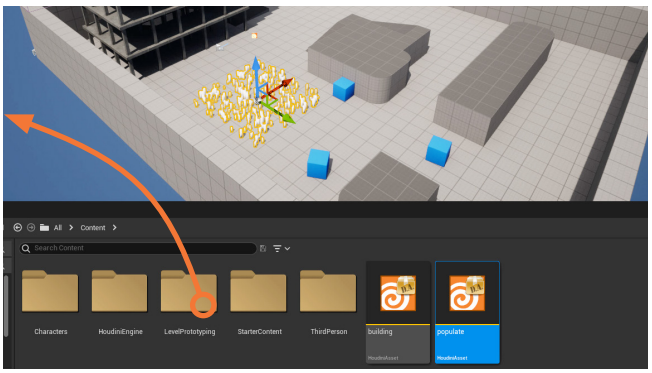
02 Set the **Operator Name** to *populate* which will change the **Operator Label** to *Populate*. Click on the button to the far right of **Save to Library.** In the *Locations* sidebar, click on $HIP and then double-click on the *hda* directory. Press **Accept** and then **Accept** again to save the asset to disk.

This creates a new Houdini Digital Asset file (.hda) on disk that is being referenced by this scene. It can also be referenced into other Houdini scenes or into other applications such as Unreal using the Houdini Engine.



03 The **Edit Type Properties** window opens up. This panel is where you will build the user interface for your asset. You will revisit this window later. Click **Accept** to close this window.

In order for you to maintain the procedural nature of the Houdini Digital Asset, you can build a high level interface that can be used to access the nodes inside the network. You will add to the interface for this asset later on in the lesson.



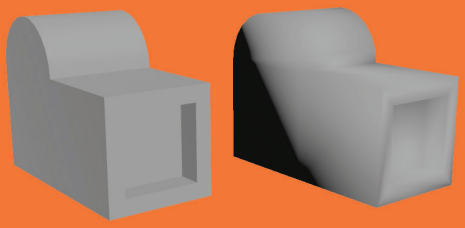
04 **IN UNREAL** – In the Content Browser, go back to the *Content* directory. Click **Import to** and find the *populate.hda* asset file in the current project directory. Drag the asset from the Content Browser to the 3D workspace.

Press **Play** and walk around the asset. It is there but there is nothing special about it. Press **Esc** to return to the asset UI.

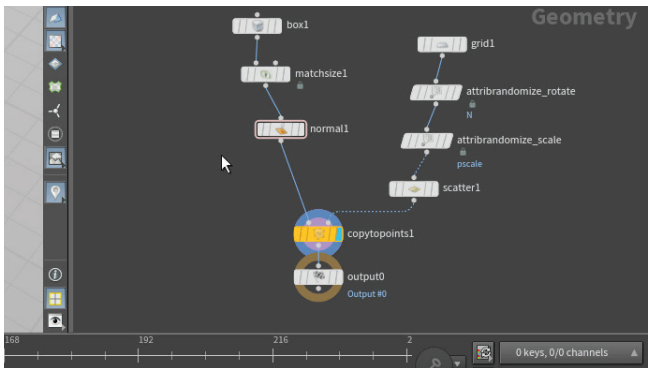
VERTEX NORMALS

By default, Houdini objects like the box have point normals but don't have vertex normals. To make sure that you have proper vertex normals, you can add the **Normal** node which uses a cusp value to decide which edges should appear hard and which ones appear soft.

Game editors such as Unreal need vertex normals to display properly which can be set up easily in the existing network.

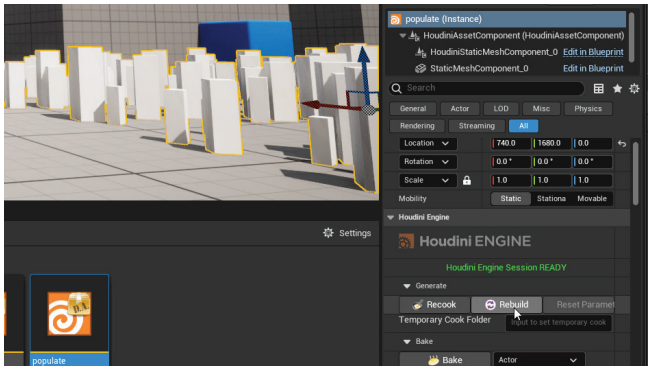


With Normals Without Normals



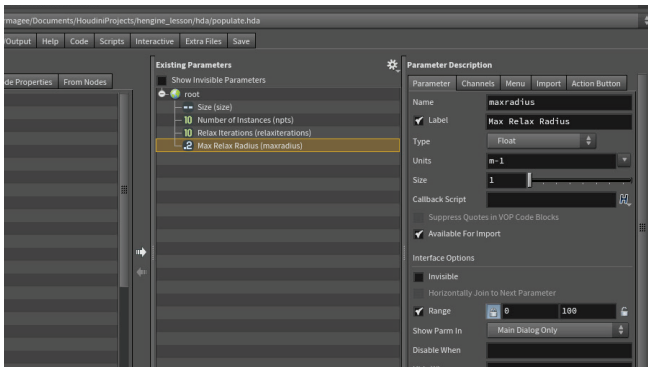
05 **IN HOUDINI** – One thing that you will notice about the asset in Unreal is that the boxes don't have sharp corners. To fix this you should go back to Houdini and in the Network editor, press **tab > "norm..."** then choose the **Normal** tool.

Add the **Normal** node right after the box node. From the **Asset** menu, choose **Save Asset > Populate**. This saves the change to the .hda file which makes the updated asset available to anyone who is has it loaded. In this case it means that you can update the asset definition inside Unreal and the correct normals will display.



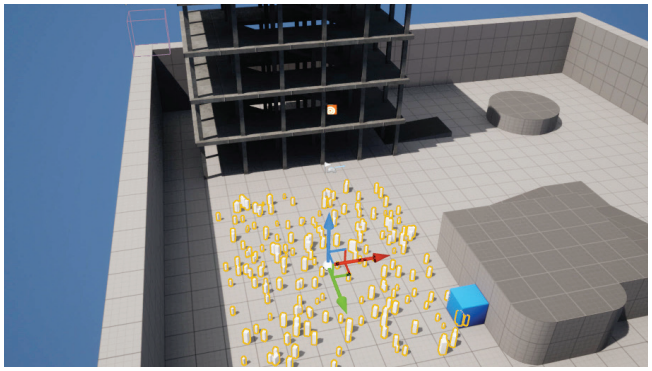
06 **IN UNREAL** – In the details panel, under Houdini Asset, open the **Cooking Actions** section. Click the **Rebuild Asset** button to accept the changes. This makes sure that the changes you made inside Houdini are properly updated in the Unreal scene.

The asset now has proper normals but you don't have any control over the procedural network. To create an interface that you can use in Houdini, you are now going to promote some parameters from inside the asset to the top level.



07 **IN HOUDINI** – Choose **Assets > Edit Asset Properties > Populate**. Click on the **Parameters** tab. In the Network editor, click on the *grid* node. From the Parameter pane, drag the **Size** parameter onto root in the **Existing parameters** list.

In the Network editor, click on the *scatter* node. From the Parameter pane, drag the **Force Total Count** parameter onto root. In the Parameter description, change the **Label** to **Number of Instances**. Drag the **Relax Iterations** and **Max Relax Radius** to add these parameters to the asset. Click **Accept** which saves these new parameters to the asset.



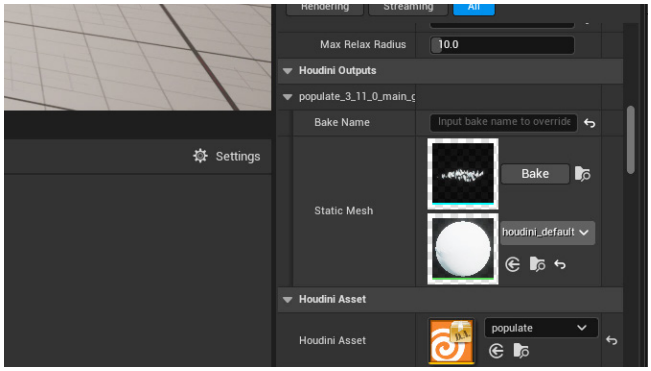
08 **IN UNREAL** – Click the **Rebuild Asset** button to accept the changes. The promoted parameters show up in the Parameter pane. Change the **Size** and **Number of Instances** to see how they affect the look of the resulting grid of boxes.

Now the procedural nature of the asset has been exposed and you can create unique versions of the asset in different levels. You can add more than one of these *populate* assets in this level and each of them can have unique settings while referencing the same asset in the .hda file. You can also use this asset in multiple levels being worked on by multiple artists.

PART FIVE

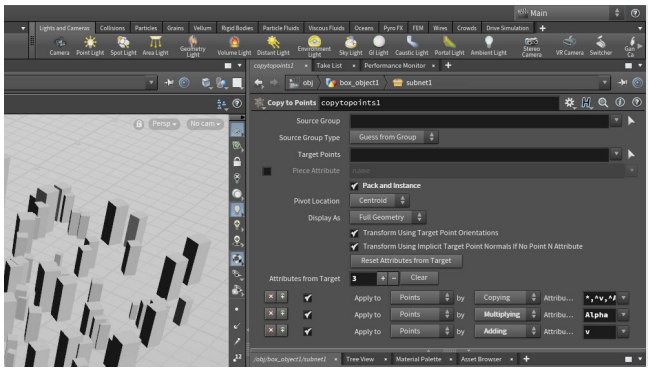
Set up Instancing

When you set up instancing in Houdini properly, you can replace the default shape you have copied to the points with other Unreal props. You can add more than one prop that will be randomly distributed in place of your default shape. You are now going to make sure that instancing is in fact set up properly and then you will use this to add some other props into the system.



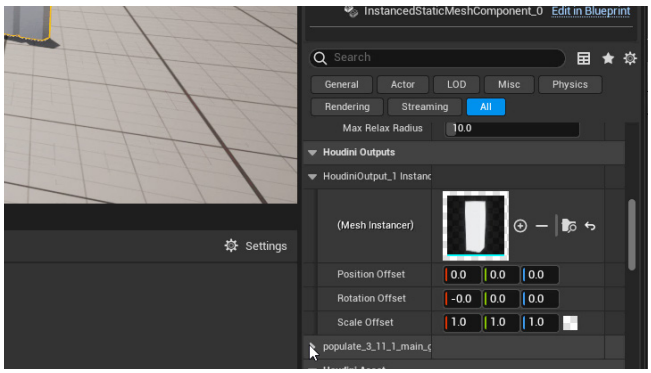
01 **IN UNREAL** – In the Details panel, go to the **Houdini Outputs** section. You can see that the whole collection of boxes is being imported as a single mesh. This means that you are not yet using instancing. Houdini is copying the boxes to the points then outputting the resulting geometry for Unreal.

This is not very efficient for gameplay. Therefore you need to set things up just a little differently to get the instancing that you need for this tool to work properly.



02 **IN HOUDINI** – Select the *copytopoints* node and turn on **Pack and Instance**. From the **Asset** menu, choose **Save Asset > Populate**.

By using packed primitives, the box geometry feeding into the *copytopoints* node is treated as a single primitive. This sets up instancing in Houdini and in turn triggers instancing in Unreal when this asset is loaded into the editor using the Houdini Engine plug-in.



03 **IN UNREAL** – Click the **Rebuild Asset** button to accept the changes. Go to the **Houdini Outputs** section. You can see that only a single box is being imported and instanced to the points. It is then rotated and scaled based on the attributes you set up earlier.

Now you can replace this default box with other geometry in your Unreal environment. This is the ideal way of populating a series of points because you have lots of flexibility in the editor to add different objects to the system.

PACKED PRIMITIVES IN HOUDINI

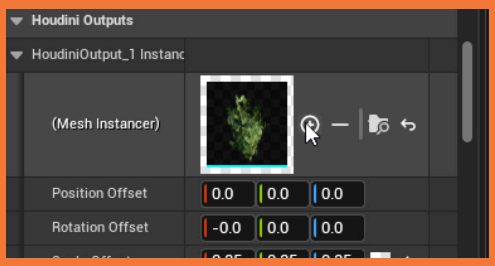
In Houdini, packed primitives provide an efficient way of managing instances for viewport display and rendering. The geometry feeding the copy node is packed into a single primitive and then treated like an instance. For the copied boxes shown in this lesson, this takes you from 1,500 primitives down to 250 once packing is in place. With the Houdini Engine plug-in for Unreal, the packed primitive is recognized as a Unreal instance which will allow for more efficient gameplay.

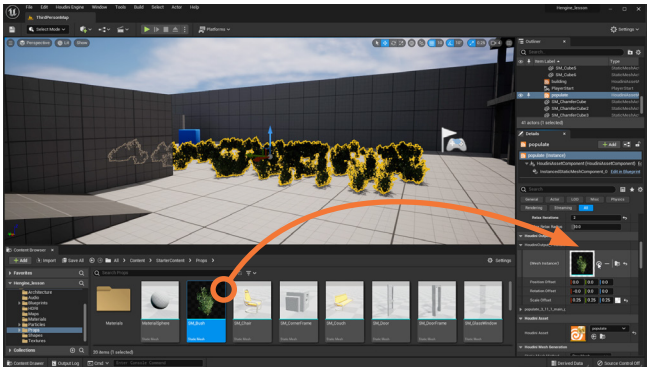
Points	2,000	Center	-0.0227511, 0.286
Primitives	1,500	Min	-3.12244,
Vertices	6,000	Max	3.07694, 0.573
Polygons	1,500	Size	6.19938, 0.573

Points	250	Center	-0.0227511,
Primitives	250	Min	-3.12244, -1.0
Vertices	250	Max	3.07694,
Packed Geos	250	Size	6.19938,

INSTANCES IN UNREAL

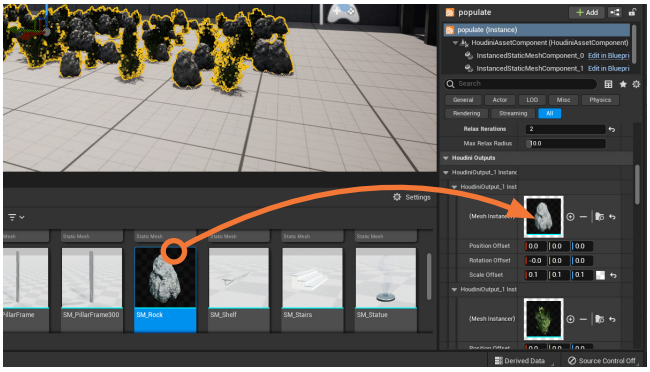
When the Houdini Engine plug-in detects packed primitives, a Houdini Output Instancer is created in the Details tab that contains the input geometry and some parameters for Rotating and Scaling the instance. You can replace this with geometry from your Unreal content window and size it accordingly. The Plus sign lets you add more instanced inputs to create more variations in the same system.





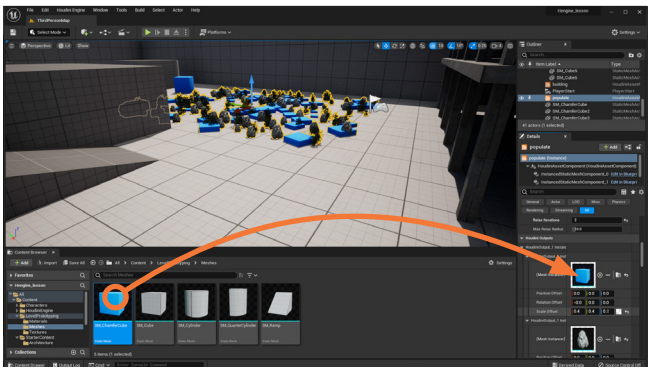
04 Go to the **Houdini Outputs** section and expand *HoudiniOutput1 instancer*. This is the box instance which you can replace with content from within Unreal.

In the Content Browser, open **Starter Content > Props**. Drag the *SM_Bush* prop over to the **Houdini Outputs**. Set the **Scale Offset** to **0.25** in all three axes. The geometry is instanced to the points in the populate asset and rotated and scaled just like the boxes.



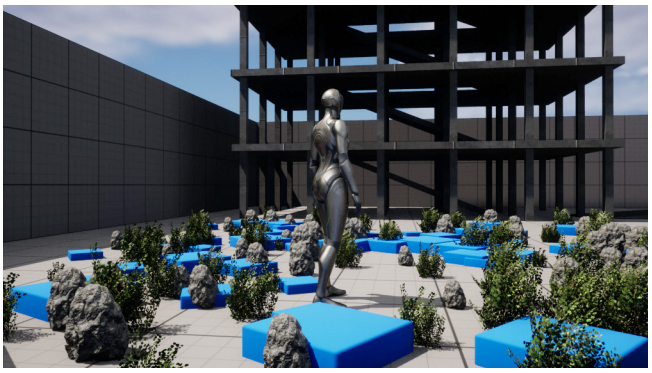
05 Now click the **Plus [+] sign** next to the instance object. This adds a second one. Drag the *SM_Rock* prop over to the new **Houdini Instanced Input**. Set the **Scale Offset** to **0.1** in all three axes..

In the Details panel, scroll to the **Houdini Engine** section and click the **Bake** button. In the Outliner, scroll to the *HoudiniAssetActor*. Click the eye icon to hide it so you can focus on the digital asset. Press **Play** and walk around the scene to see the instances in action.



06 Now click the **Plus sign** next to the bush object. This adds a third one. Navigate to the **Content > LevelPrototyping > Meshes** folder. Drag the *SM_ChamferCube* over to the new **Houdini Instanced Input**. Set the **Scale Offset** to **0.4 0.4 0.2.**

You can continue to add more instanced objects to create more variety and maybe change the size. You can use the **Relax Iterations** parameter if you need to separate the instances from each other a bit more.



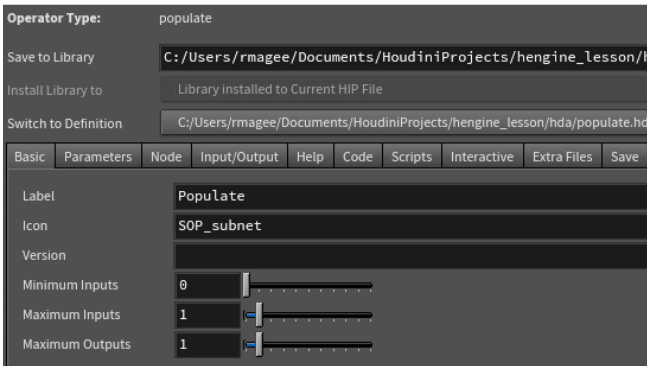
07 In the Generate section of the *building* asset's **Details** panel, press **Rebuild**. In the Bake section, turn on **Replace Preview Bake** then click on the **Bake** button.

Press **Play** to walk around the scene to see the instanced geometry in action. Now you can see that you are colliding with the cubes which already had collision geometry set up properly. You should always make sure your instanced objects have collision geometry set up properly.

PART SIX

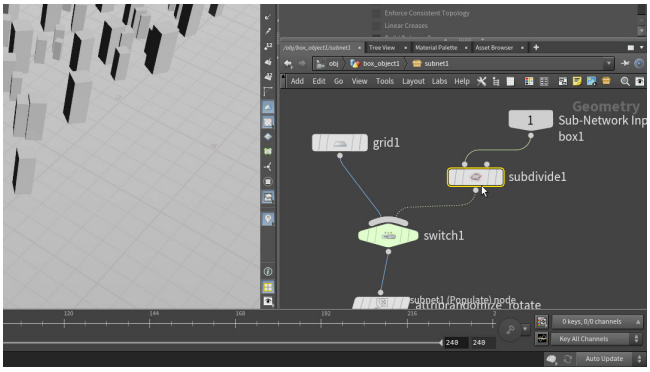
Use Geometry to Drive Asset

So far this asset supplies the input geometry for the populate asset. Another option is to use existing geometry in the Unreal scene to instance the geometry. You will now add an input node to your asset which will accept this Unreal geometry. The ability to work with objects on our level creates better integration between your procedural assets and existing game art.



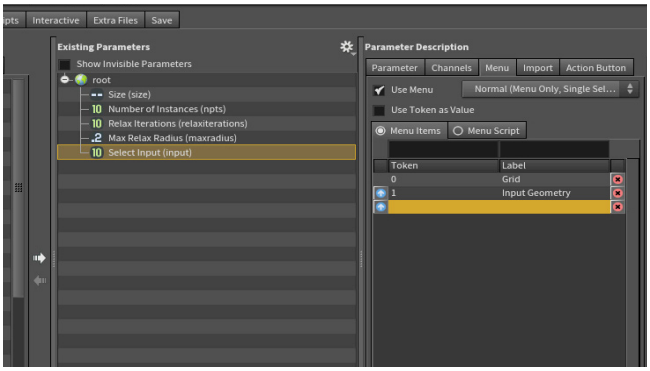
01 **IN HOUDINI** – Choose **Assets > Edit Asset Properties > Populate**. Stay on the **Basic** tab and set **Maximum Inputs** to **1**. Click **Accept** on the Type Properties window to save the changes. This will create an input node inside the asset that you can wire into the network. This input will let you grab geometry from the Unreal scene later.

Make sure you leave **Minimum Inputs** set to **0**. If it is set higher than that then the asset will ONLY work if the minimum input requirement is met. Otherwise the asset won't cook and you won't see anything.



02 In the Network editor, add a **Switch** node after the grid and wire in the new Input. Add a **Subdivide** node then wire it after the input node to make sure there is enough detail for the randomizing of attribute values. Set **Depth** to **5**.

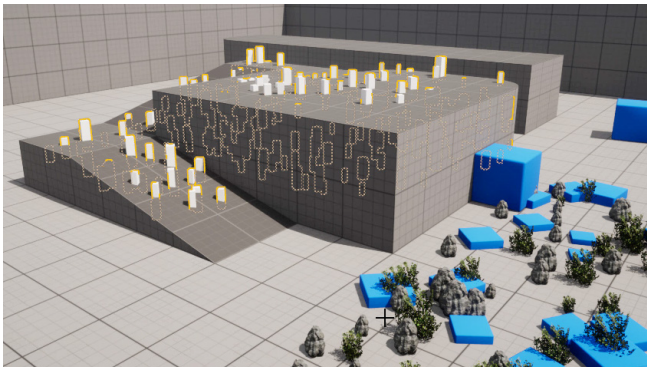
You are getting an **Error** on the subdivide node because in Houdini there is nothing feeding into the third input. Go back up one level and feed a **box** into the asset's input to help preview how this is going to work. The box will subdivide into a sphere in Houdini which is not how it will work in the final scene.



03 Choose **Assets > Edit Asset Properties > Populate**. From the *switch* node, drag the **Select Input** parameter to the Parameter list. Select the **Select Input** parameter. Click on the **Menu** tab then turn on **Use Menu**.

Add **0, Grid** then **1, Input Geometry**. Click **Accept** on the **Type Properties** window to save the changes.

This will add a menu to the UI of the asset which you can use in Unreal.



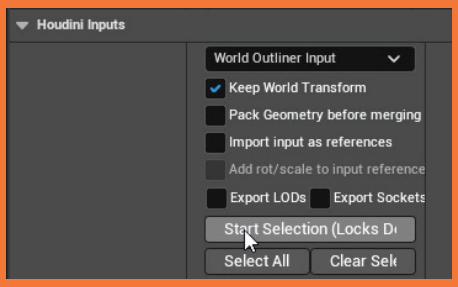
04 **IN UNREAL** – Click the **Rebuild Asset** button to accept the changes. Drag a new *populate* asset into the foreground. Go to the **Houdini parameters** section and from the **Select Input** menu choose **Input Geometry**.

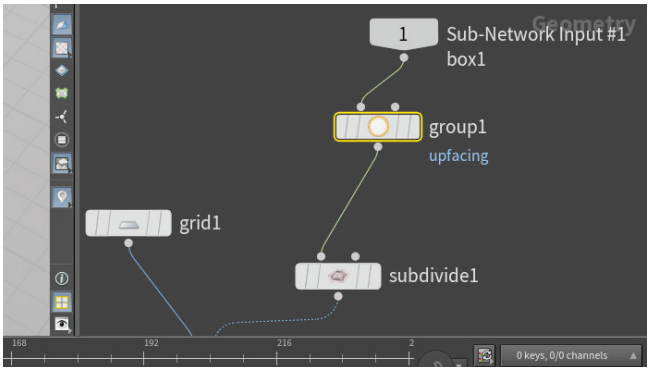
In the **Details** panel, go to the **Houdini Inputs** section and from the menu, choose **World Outliner Input**. Click on the **Start Selection** button and in the 3D scene, select all the parts of the ramp and platform. Click **Use Current Selection** and now the instances are being scattered inside the subdivide platform. This is not exactly what we need.

USING THE INPUT NODE IN UNREAL

When you set up an input node in your asset, there are a number of options for accessing input geometry. You can use Geometry from the content browser.

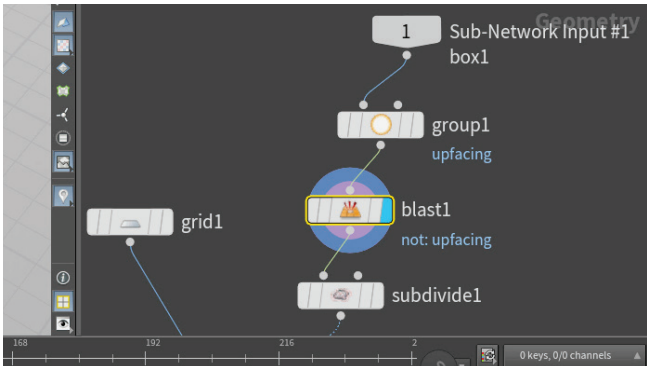
You can set up **Curve Input** where you draw a curve in Unreal. You can also grab content from the **World outliner** or input **Unreal landscapes** at heightfields for use with Houdini's new terrain toolset.





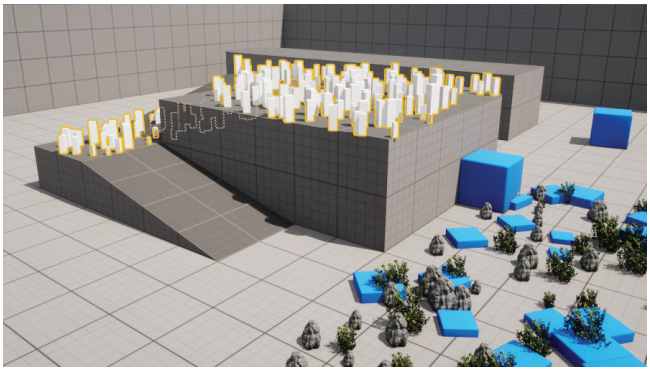
05 **IN HOUDINI** –You are now going to isolate the top faces of the box to limit where the points are being copied. Insert a **Group** node after the *input* node in the network editor. Set the **Group Name** to **upfacing**. Under **Base Group**, turn **Enable** to **Off**. Under **Keep by Normals**, turn **Enable** to **On**. Set the **Direction** to **0, 1, 0** and **Spread Angle** to **0**.

Instead of scattering points to all the faces of the geometry in Unreal, you are going to isolate the top faces and use them instead.



06 Insert a **Blast** node after the *group* node and set the **Group** to *upfacing* and turn **Delete Non Selected** to **On**. Now only those primitives facing up will be kept - the others will be deleted. Choose **Assets > Save Asset > Populate**.

The key here is that because you used a group to identify the top faces, it doesn't matter what geometry is input into the asset the solution will work. This is how a procedural asset works because it provides a generalized solution instead of a specific one-off solution.



07 **IN UNREAL** – Click the **Rebuild Asset** button to accept the changes. Now only the top faces of the geometry have boxes on them.

This asset can now use a default grid or geometry sourced from the level to work. There are always lots of options available when building Houdini Digital Assets for use in Unreal.

08 Set the **Number of Instances** to **40**. Go to the **Houdini Outputs** section and expand the instanced box.

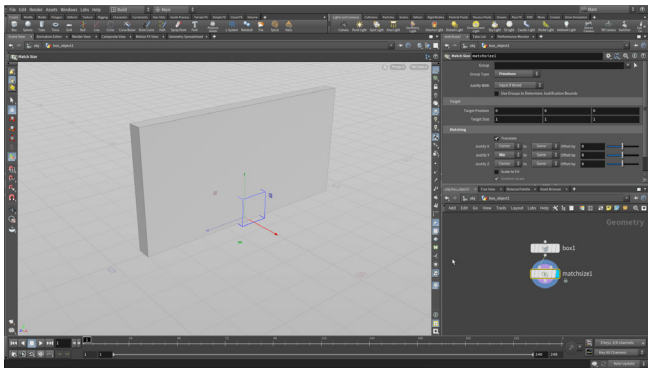
In the Content Browser, open **StarterContent > Particles**. Drag the *P_Fire* prop over to the **Houdini Instanced Input**. Set the **Rotate Y** to **90**. Press **Play** to test the level.

The instanced points in this asset can be used for more than just geometry. They are now a part of the Unreal level and can be used to solve different problems. You may want to hide *populate2* with the fire while you add more to your scene. It will still appear when you play the level.

PART SEVEN

Import RBD Simulation into Unreal

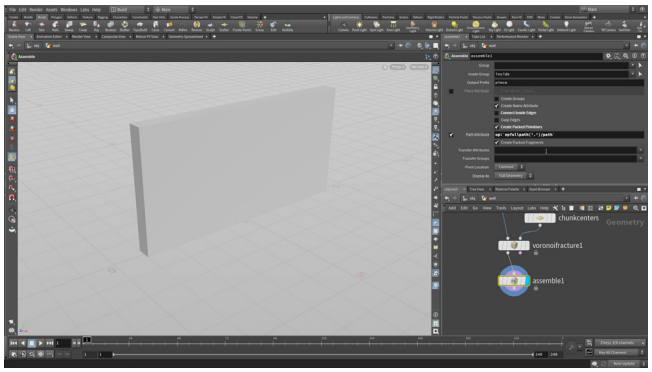
In this part of the lesson, you are going to create a wall then smash it using rigid body dynamics. Export this system to FBX then import into Unreal for use in your game. This is a simple example of bringing visual effects from Houdini to Unreal.



01

IN HOUDINI - Create a **Box**. Go to the geometry level and set its **Size** to **0.5, 4, 8.**

Add a **Match Size** node and in the Parameter pane, under **Matching**, set **Justify Y** to **Min**. This raises the box up so that it sits on the ground.



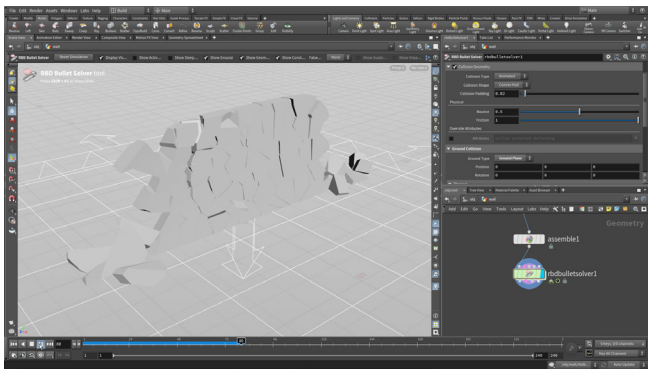
02

Go to the Object level. Rename the node *wall*. Select the wall node and from the **Model** shelf, select **Shatter**.

Dive down to the geometry level, select the *chunkcenters* node and set the **Force Total Count** to **100**.

Add an Assemble node at the end of the chain. Turn off **Connect Inside Edges** and turn on **Create Packed Primitives** and **Path Attribute.** Change the **Path Attribute** to:

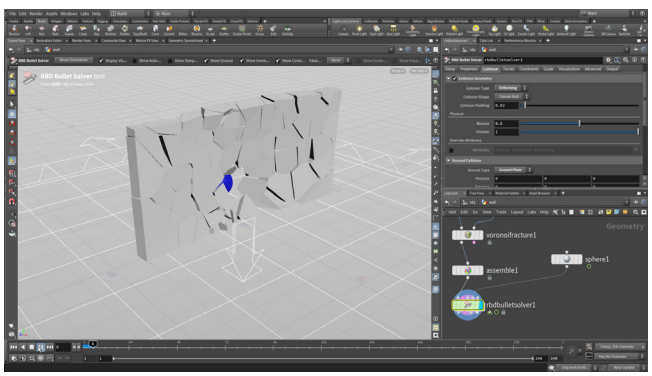
```
op:`opfullpath('.')/path'
```



03

Add an **RBD Bullet Solver** node to the end of the chain. On the **Ground** tab, set **Add Ground Plane** to **Ground Plane**.

Press **Play** to run the simulation. Not much happens because the pieces of the wall are just falling down.



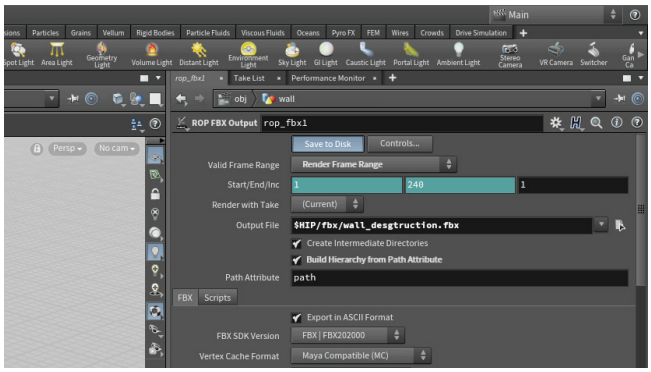
04

Add a **sphere** node to the network. Set it in front the wall with a **Center X** of 2 and a **Center Y** of **1**. Press the **Alt** key and click on **Center X** to keyframe it.

Go to **frame 9.** Set **Center X** to **-3** and **Alt-click** on **Center X** to set a second keyframe.

Wire the sphere into the **fourth** collision input of the *rbdbulletsolver* node. Under **Collisions** set **Collision Type** to **Deforming.**

Press **Play** to run the simulation. The wall is now being smashed by the sphere which will be hidden in the simulation.

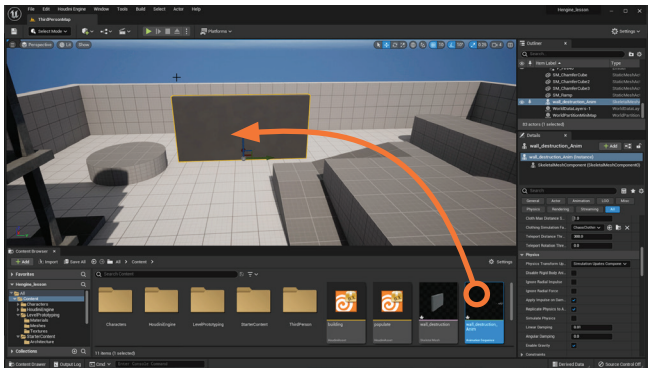


05 Add a **Transform** node after the *rbdbulletsolver* node then set **Uniform Scale** to **100**. This scales the simulation to the size needed in Unreal.

Add an **FBX Output** node. Set **Valid Frame Range** to **Render Frame Range** then set **Output File** to *$HIP/wall_destruction.fbx*.

Turn on **Build Hierarchy from Path Attribute.**

Click **Save to Disk** to save to disk.



06 **IN UNREAL** – In the Content Browser, go back to the *Content* directory. Press **Import** and grab the *wall_destruction.fbx* file. Check **Skeletal Mesh, Import Mesh, Import Animations**.

Expand the Animation section and make sure **Import Meshes in Bone is checked off.** Expand the **Mesh** section and set **Normal Import Method** to import **Normals and Tangents**. Click **Import**.

A panel will come up saying that UVs haven't been set up yet. Close this. Four new items appear in the content list.

Drag the *wall_destruction_Anim* asset into your workspace.

07 Press **Play** to see the simulation working in game. Right now the animation just loops and doesn't interact with anything. You could set up a Blueprint to trigger the animation based on some action on the character's part.

CONCLUSION

You have now created a variety of game assets for use in Unreal. From Houdini Digital Assets to FBX files containing rigid body simulations, you now have a good foundation to build from. These Digital Assets let you integrate Houdini's node based workflow inside host applications such as Unreal. The same workflow works with other applications such as Unity, Autodesk Maya and Autodesk 3DS Max.

Go to **SideFX.com/unreal** for more information. This will point you to a starter kit of assets you can use in Unreal right away along with a link to more tutorials.

Be sure to take a look at **Project Titan** an in-house tech demo designed to explore the creation of a 3D environment that leverages the latest technologies in Unreal. The tools and techniques created for Project Titan have been shared with the community as learning materials and downloadable content.

HOUDINI FOUNDATIONS

BUILD A CITY WITH PDG

To manage pipeline workflows, you can turn to Task Operators (TOPs) which are built using a technology called the Procedural Dependency Graph (PDG). Workflows created using PDG use TOP nodes that generate work items that distribute tasks to either your local computer or a larger compute farm.

The TOP network lets you determine the dependencies between different work items and how they are contributing to the final output. This information is easy to visualize in the node graph which can be used to define how you want data to flow through your network. TOPs lets you build workflows that can be used to automate, analyze and scale your pipeline.

In this lesson, you will use TOP nodes to take a city map, create buildings for each city block and then grow this system to handle more complex buildings and larger city maps. Houdini artists will probably know how to do this in SOPs, but by using TOPs, you can learn the PDG workflow while creating a system that can be easily scaled to distribute multiple tasks to an external compute farm to process in parallel.

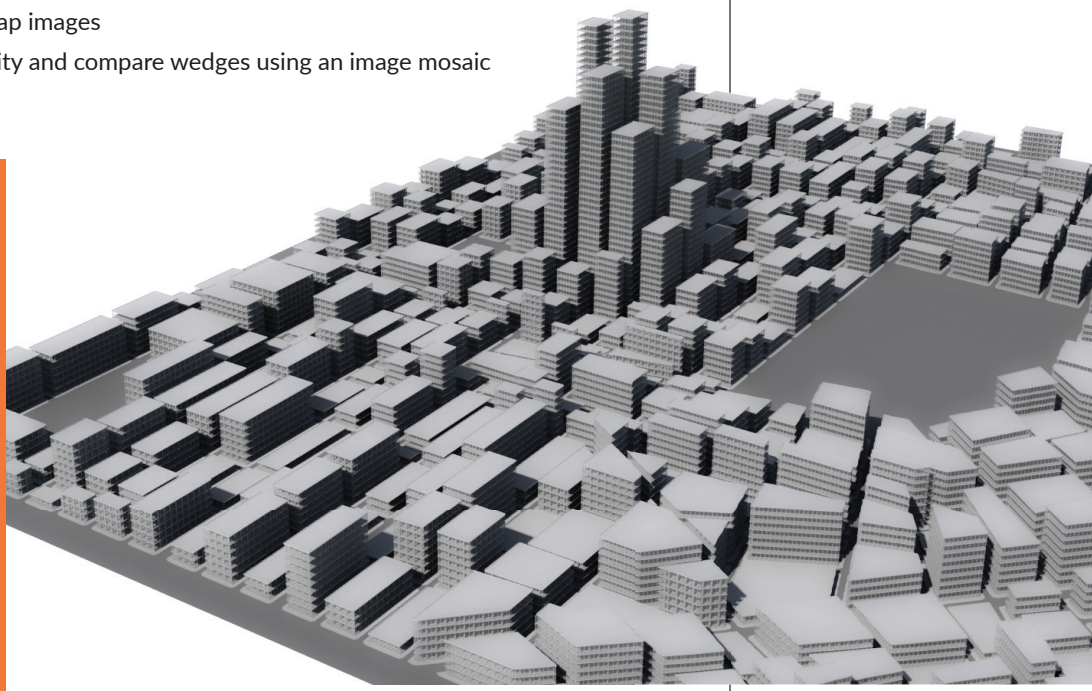
NOTE: *This lesson uses Image Magick - make sure this app is installed on your computer.*

LESSON GOAL

- *To create a TOP (Task Operator) Network to build up a procedural city and then render it out.*

WHAT YOU WILL LEARN

- How to convert a city map image into geometry
- How to set up a TOP network to save out the city blocks
- How to create Geometry in TOPS to build buildings on each city block.
- How to create a city core to make some buildings higher than others
- How to wedge the cityscape to try different locations for the city core
- How to wedge the use of different map images
- How to use TOPs to render out the city and compare wedges using an image mosaic



LESSON COMPATIBILITY

Written for the features in Houdini 19.5+

The steps in this lesson can be completed using the following Houdini Products:

Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

Document Version 2.0 | Oct 2022

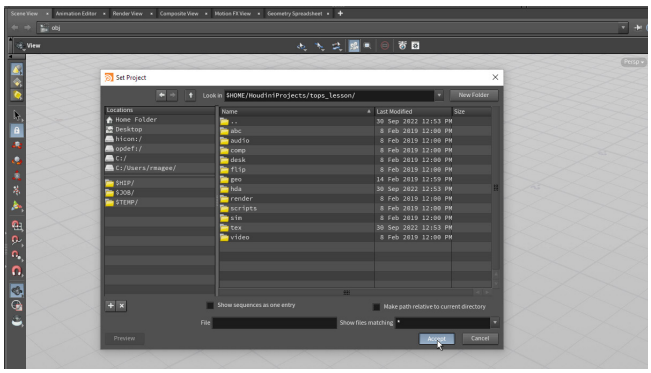
PART ONE

Create a City Grid

To build a procedural city, you will start with a city grid. You are going to create the geometry by tracing an image file that includes a black and white image of a city map. This will be the input geometry for the network you will be building in TOPs.

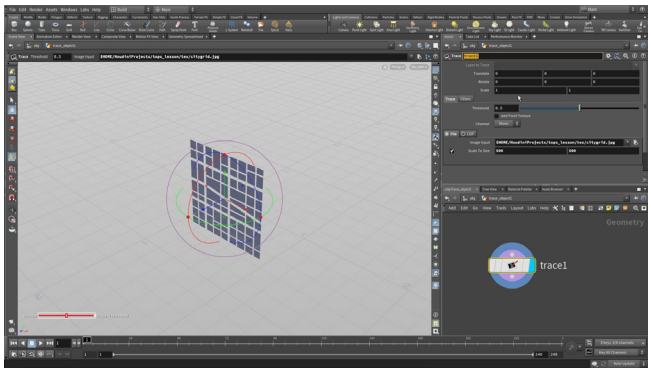
PROJECT FILES

Go to the **Foundations tutorial** page on **SideFX.com**, where you got this tutorial, and download the *tops_lesson* directory. Put it into the **Houdini Projects** directory which you can find in either the **home** directory or the **documents** directory.



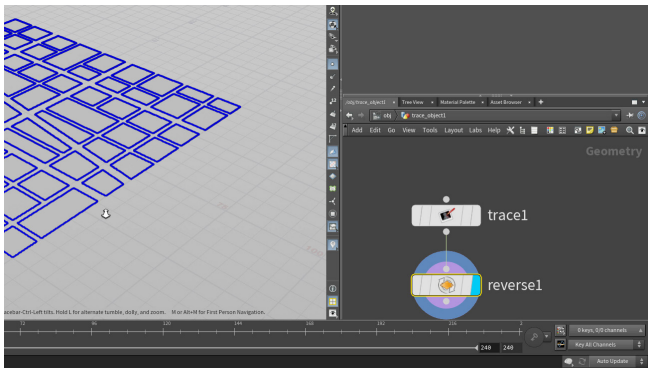
01 Select **File > Set Project.** Find the *tops_lesson* directory that you downloaded and press **Accept**. This makes this project directory and its sub folders the place for all the files associated with this shot.

Select **File > Save As...** You should be looking into the new *tops_lesson* directory. If not then click on **$JOB** in the left hand column. Set the file name to *city_01.hip* and click **Accept** to save.



02 In the viewport, **press tab** to bring up a menu and start typing **TRACE**. Choose **Trace** and your cursor now shows the outline of a square waiting to be placed in the scene. **Press Enter** to place it at the origin. Right now it is a traced circle.

Double-click on the node in the Network view to dive down to the geometry level. Select the *trace* node and click on the **File chooser** next to the **Image Input** parameter. Click on **$HIP** then navigate into the *tex* directory. Select the *citygrid.jpg* image and then click **Accept**. Next turn **ON** the **Scale to Size** option and set it to **500, 500**. This will give you more accuracy.

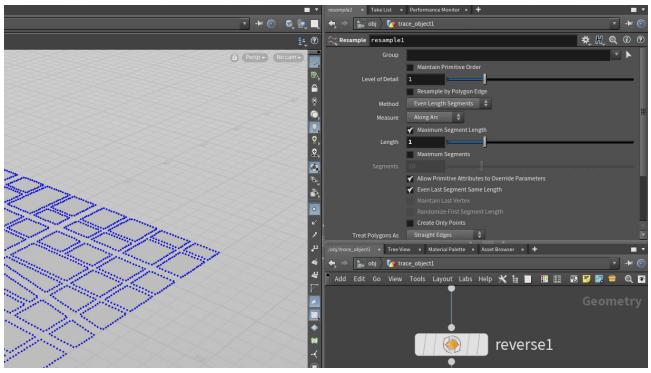


03 In the upper section of the *trace* node, set the following:

- **Rotate X** to **-90**
- **Scale** to **100, 100**

In the Network view, press **tab** and start typing **Reverse.** Choose the **Reverse** node and place it down then wire the *trace* node into it. Set its **Display Flag**. This node will point the normals up. In the viewport, press **Spacebar-H** to view the whole city grid.

Turn on **Display Points** in the **Display option** bar to see that there are lots of trace points around each city block.



04 In the Network view, press **tab > Resample** and click to place it under the *reverse* node. Don't wire it in yet. Set **Length** to **1** then connect the output of the *reverse* node into the *resample* node's input.

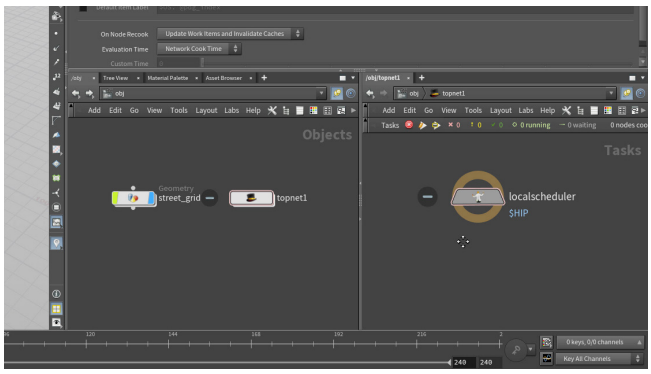
RMB-click on the output of the *resample* node and choose **Null**. Click to place the null node at the end of the chain. Set its **Display flag** and rename it *CITYBLOCKS_OUT*.

Save your work.

PART TWO

Generate and Display Work Items

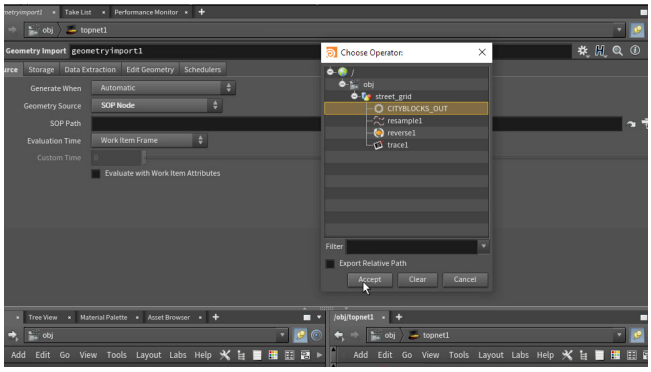
Now that you have a city grid, you can break out the different city blocks into separate work items. That will allow you to use each block to generate buildings. You will set up a TOP network to save out each of the blocks. At the same time, you will create an object for visualizing selected work items to verify that you are getting the results you want.



01

Go back to the object level and rename the object to *street_grid*. Go to the Network view and **press Tab** and start typing **TOP...** and select **TOP Network**. Click to place the node in the network.

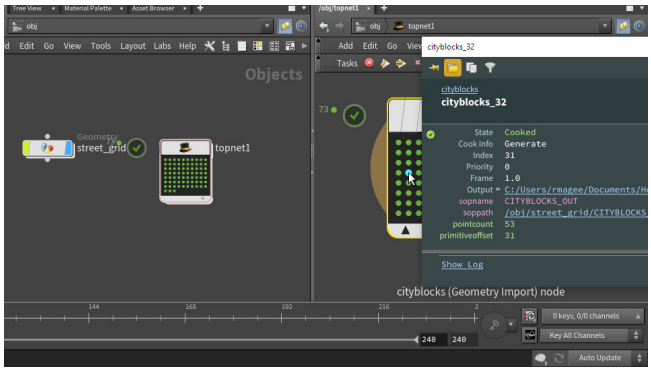
Click on the little **arrow** in the top right of the Network View. From the menu choose **Split Pane Left/Right.** Click on the **Pin** icon on the Network view to the left. The other Network view is already pinned. In the Network view on the right, **double-click** on the *topnet* to dive into it. You can now work with both networks as you develop the city.



02

In the TOP network press **tab > Geometry Import**. Click to place the node. Set **Geometry Source** to **SOP Node** and then click on the **Choose Operator** icon next to **SOP Path.** Use the floating window to navigate to and select the *CITYBLOCKS_OUT* null node.

Under **Storage**, leave **Store Geometry As** set to **External File**. Under Data Extraction, set **Copy from Class** to **Primitive**. This will store the files to disk so that they can be retrieved by the next TOP node. This is especially important if you set up a TOP network to distribute tasks to a compute farm.



03

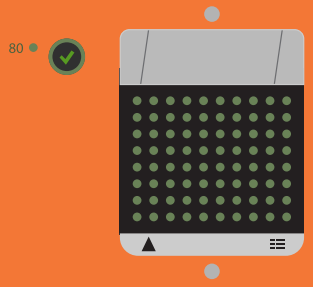
Rename the *geometryimport* node to *cityblocks* and from the **Task bar** at the top, click on the **Cook Selected Node** button. You could also select it and hit **Shift-G** to cook it. You will see dots representing work items appear as the node is cooked.

Ctrl-MMB-click on the dots to view the work item attributes. Some attributes like **Index** are basic tops attributes that every work item will have. Others are specific to the type of work item. You will see that each work item is associated with an **Output** file. You can **RMB-click** on a work item and choose **View Work Item Output** to see the geometry it contains in a separate geometry viewer.

WORK ITEMS

TOP nodes create work items for each task that you ask it to perform. These are represented on the TOP node as dots and you can use them to visualize their status and can select them individually to evaluate how that work item is progressing.

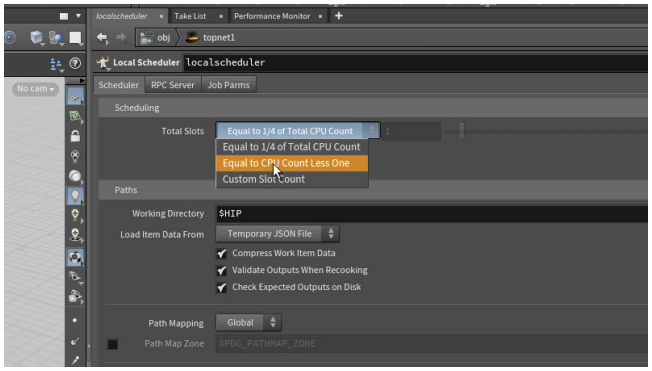
When all the work items are fully cooked, the node gets a checkmark and the completed work items are colored green.



WHAT IS THE SCHEDULER NODE?

When you first create a TOP network, a local scheduler node is created to organize tasks as they are cooked. The local scheduler points to your local computer and will use a percentage of available cores to cook. You can set **Total Slots** to **Equal to CPU Count Less One** option to match available cores.

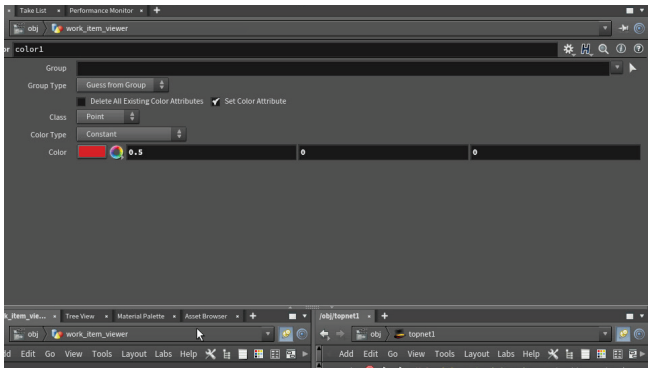
You can also set up scheduler nodes for **HQueue, Deadline, Tractor** and **Python** to send tasks to a larger compute farm. This will allow more work items to be processed in parallel to make your graph more efficient.



04 If you want to speed up processing on your network, select the *localscheduler* node and set **Total Slots** to **Equal to CPU Count Less One.** Now future cooks will be faster because more processors are being used.

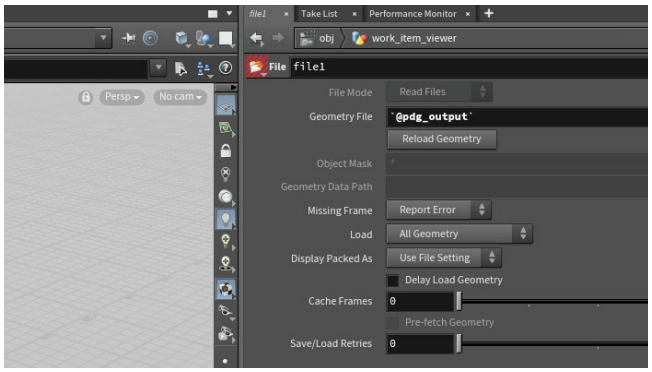
To see the work items in the viewport, you will need to set up a work item viewer. In the Network view on the left, **press tab > File**. Click to place the node and rename it *work_item_viewer*.

Normally you would point a File node to a file on disk. You will do this at first to find the files then use a different method to acquire the work item directly from TOPS.



05 Double-click to dive inside the *work_item_viewer* node and in the Parameter pane, click on the **Geometry File** file selector button. In the File selector, click on **$HIP** and then **double click** on the *geo* folder.

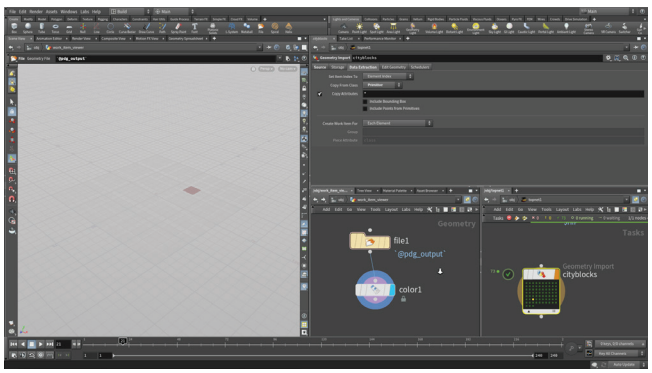
Select the *city_01_cityblocks* file sequence. This imports the saved out geometry as a numbered sequence. Add a **color** node after the geometry sequence and set its color to **red (1, 0, 0)**. Scrub in the timeline to see the different pieces of geometry loading in sequence. There are 73 pieces that have been saved out to disk.



06 Instead of linking the display of the city blocks to the frame number, link it directly to the top network. Click on the **File** node and change **Geometry File** to the following expression:

```
`@pdg_output`
```

This means that instead of loading the files from disk in sequence, you will load whichever work item is active in the topnet. At first, you get an error because there aren't any work items being output from the PDG network.



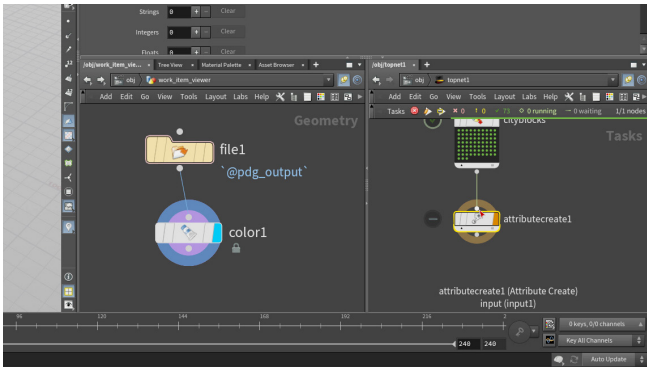
07 From the **Active Work Item** menu in the Geometry Network view, select different work items to select them in the Scene view. The red color moves to the selected geometry. The geometry related to that work item will appear in the viewport. In the topnet, you will see that dot on the *cityblocks* TOP node highlighted in yellow.

Save your work.

PART THREE

Add Attributes

To create buildings, you want to set a fixed base height and a random height variation so that your buildings are not all the same size. You could set up these attributes at the geometry level of the city map but you can also assign them here in TOPs. This will make it easier to make changes at the TOP level if needed down the line.



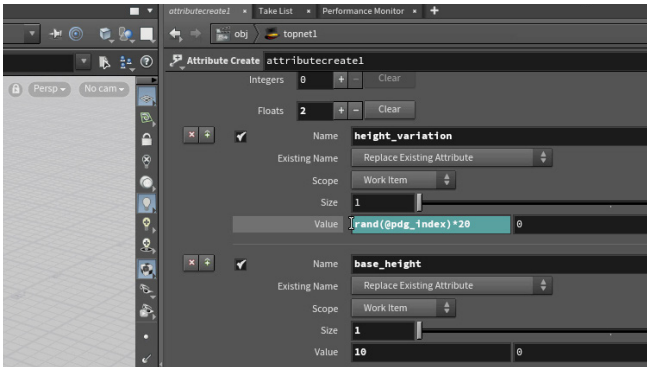
01

In the TOP net, add an **Attribute Create** TOP. Set **Generate When** to **Each Upstream Item is Cooked**.

Click on the **Plus** sign under **Float Attributes** to create a new attribute then enter the following:

- **Name** to *base_height*
- **Value** to **10**

This will ensure that all the buildings will have a minimum value of 10. You can adjust this number later if you want the lowest buildings to be a different height.

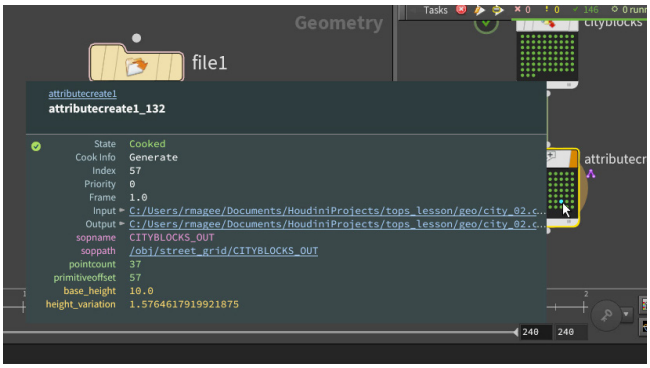


02

Click on the **Plus** sign again and enter the following:

- Name to *height_variation*
- **Value** to `rand(@pdg_index)*20`

This will create a random number from 0 to 20 using the work item attribute **Index** as the seed. When you build the network for creating a building, you will add this value to the base height value to determine how tall each building will be.



03

Select the *attributecreate* node and press **Shift-G** to cook it. In the 3D View, you will still see footprints because you haven't created buildings yet.

If you **MMB-click** on the work items, you can see that each one has a *base_height* of **10** and a *height_variation* that is a number somewhere between **0 and 20**. The attributes have been generated in this TOP node but they aren't being used yet.

ADDING ATTRIBUTES IN TOPS

It would have been possible to add attributes in the geometry network of the city grid but adding them in TOPs makes it easier to make changes to them within the context of TOPs.

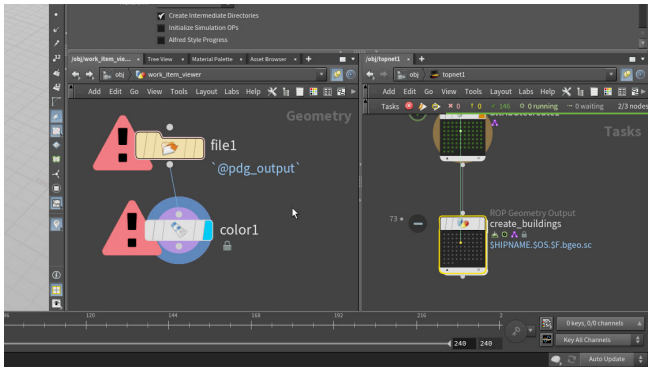
Attributes are an important way of feeding important information through your pipeline and setting them up properly is important.

```
pointcount        29
primitiveoffset   26
base_height       10.0
height_variation  19.69480037689209
```

PART FOUR

Create Buildings for the City Grid

You are now going to use a ROP Geometry node to create a simple building. To see what you are doing in the viewport, you need to generate the work items and select one of them. You can then use that work item to design the building at the geometry level.

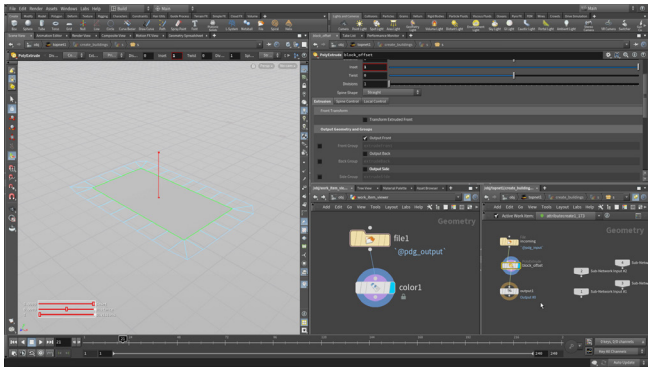


01

Add a **ROP Geometry Output** TOP. Rename it *create_buildings*. Turn **OFF** the **Use External SOP** option.

RMB-click on it and choose **Generate Node** to create the work items. These are grey work items that have not been processed yet. In effect they are place holders for the tasks that will take place once you set up this node.

IMPORTANT: Click on one of the work items. This will produce an error in the *work_item_viewer* network because this node isn't producing any geometry yet.

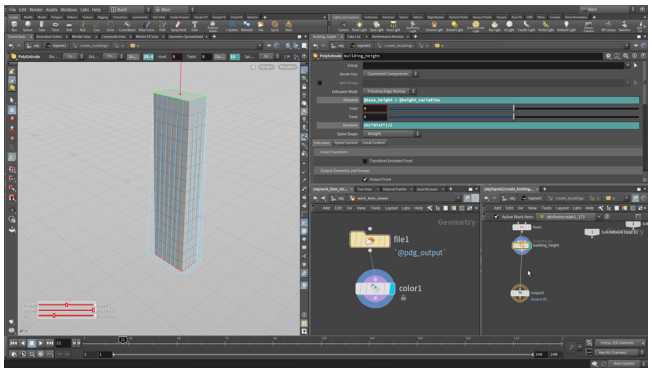


02

Double-click on this node to dive to the geometry level.

Create a **PolyExtrude** node and place it between the *incoming* and *output* nodes. Rename it *block_offset*. Set the **Inset** to 1 and under **Extrusion**, turn **OFF** the **Output Side** option. This will inset the buildings to allow for sidewalks.

Next, add a **Fuse** node after the *polyextrude* node and set **Snap Distance** to **0.2** to remove any lines that are small. Note that you are seeing something in the viewport even though the *work_item_viewer* nodes are showing errors. You will fix this later.



03

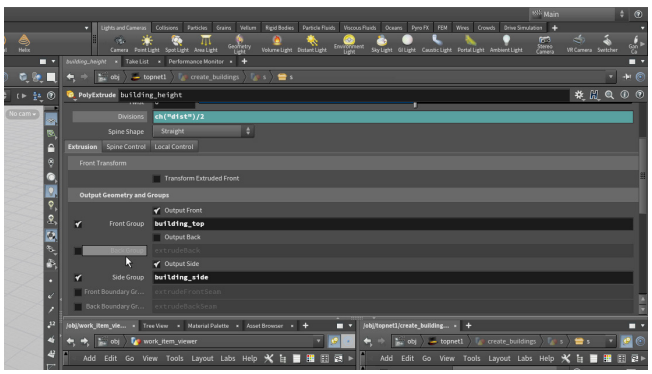
Create a second **PolyExtrude** node after the *fuse*. Rename it to *building_height* and set the **Distance** to:

```
@base_height + @height_variation
```

Make sure the **Distance** parameter is highlighted. **RMB-click** on the parameter and choose **Copy Parameter**. **RMB-click** on the **Divisions** parameter and choose **Paste Relative References**. Edit the resulting channel reference by dividing it by 2:

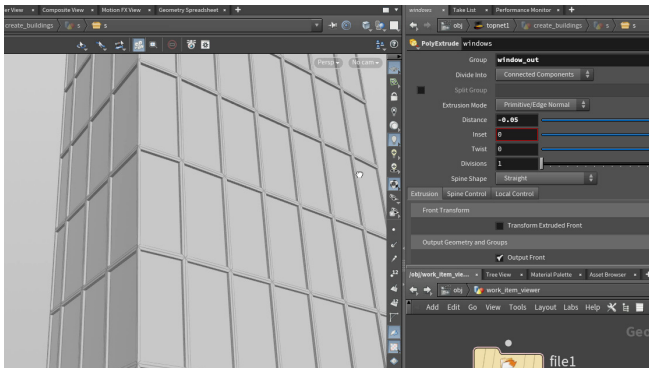
```
ch("dist")/2
```

Now you can see a grid of floors and windows.



04

In the **Extrusion** tab, turn **ON** the **Front Group** and name it *building_top* and then turn **ON** the **Side Group** and name it *building_side*. You will use these to add detail to the building.

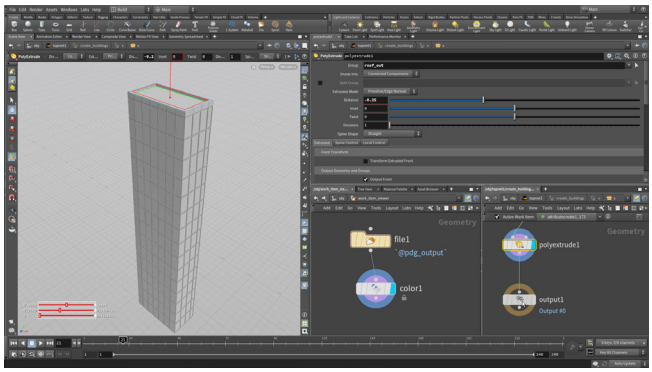


05

Add a new **Poly Extrude** node into the chain and rename it *window_frames*. Set the following:

- **Group** to *building_side*
- **Inset** to 0.05
- **Divide Into** to **Individual Elements**
- In the **Extrusion** tab, turn **ON** the **Front Group** and name it *window_out*.

Add another new **Poly Extrude** node and rename it *windows*. Set **Group** to *window_out* and **Distance** to **-0.05**.



06

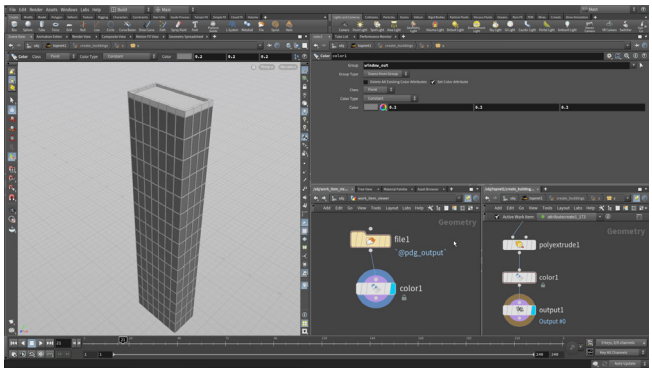
Add a new **Poly Extrude** and rename it *extend_roof*. Set

- **Group** to *building_top*
- Distance to **0.5**

Add another **Poly Extrude** node and rename it *roof_edge*. Set

- **Group** to *building_top*
- Inset to **0.3**
- In the **Extrusion** tab, turn **ON** the **Front Group** and name it *roof_out*.

Add another new **Poly Extrude** node into the chain and rename it *roof*. Set **Group** to *roof_out* and **Distance** to **-0.25**.

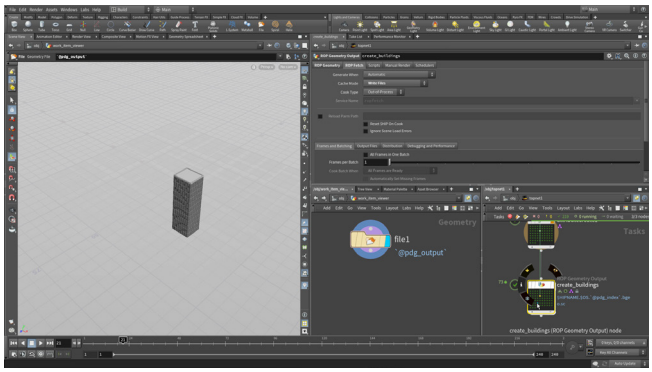


07

Add a color node after this and set the following:

- **Group** to *windows_out*
- **Color** to a **dark grey (0.2, 0.2, 0.2).**

Make sure that these nodes are wired into the *output* node and set the **Display flag** on that node. Now the frames will be a light color and the windows will be darker. This will help you read them in the viewport.



08

Go back up one level and on the *create_buildings* TOP change `$F` to `` `@pdg_index` `` to set the **Output File** to:

```
$HIP/geo/$HIPNAME.$OS.`@pdg_index`.bgeo.sc
```

This will use the index file of each building instead of the frame. Click on the **ROP Fetch** tab and set the **Cache Mode** to **Write Files.**

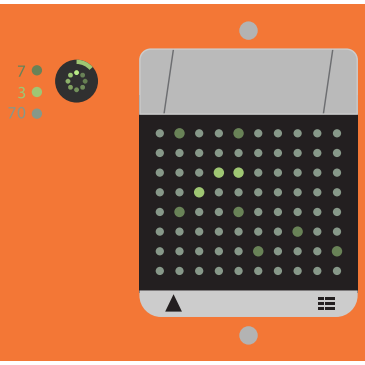
Save your scene file then select the *create_buildings* TOP and press **Shift-G** to cook it.

Go back inside the *work_item_viewer* and remove the color node so you no longer see red. Now on the *create_buildings* TOP node, click on the work items to see the buildings with different heights.

WORK ITEM PROGRESS

Up until now the work items have processed very quickly but this node takes a little longer. You can see the progress wheel which shows the completed tasks in dark green, the in-progress tasks in yellow and the queued work items in grey. This lets you observe the node as it completes the assigned tasks.

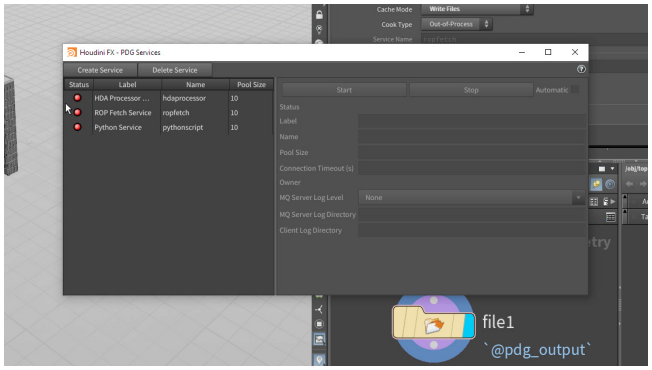
At the Object level, the TOP Network also displays work item progress for all the tasks in the network.



PART FIVE

Combine the Buildings

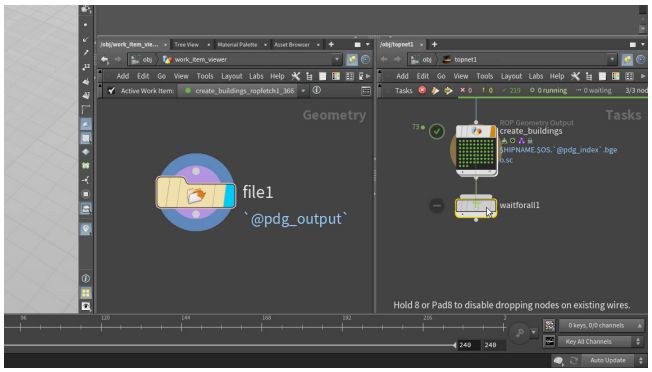
Once the buildings have been completed, you will want to bring them back together to create the city. This involves a partition node called Wait for All and a Geometry Import. You will also add in a city core where the buildings will be taller than elsewhere in the city.



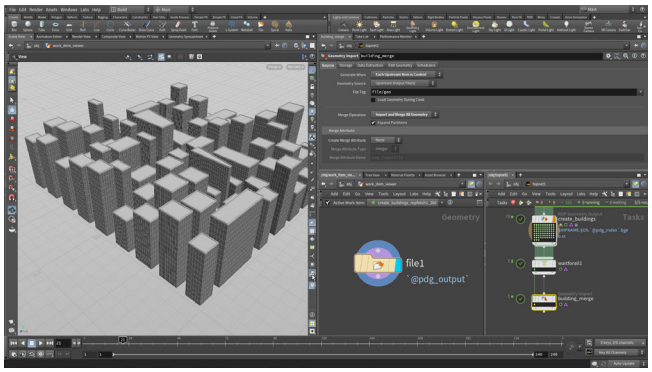
01 To speed up the processing of the tasks, you will use something called PDG Services. In the TOP Network view select **Tasks > PDG Services**. Here you can see the service set up and ready for you to use.

Close the Services window then go to the **ROP Fetch** tab of the *create_buildings* TOP and set **Cook Type** to **Services**. This will speed up the processing of this TOP node.

Save your scene file then select the *create_buildings* TOP and press **Shift-D** to dirty it then **Shift-G** to cook it. You will see that it is much faster than it was the last time.

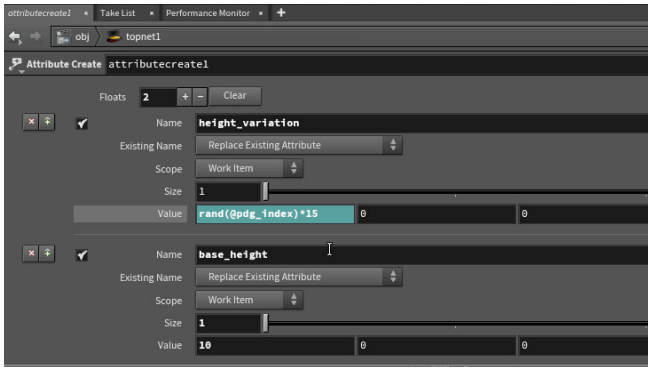


02 Add a **Wait for All** TOP node. This will just take all the work items that the ROP Geometry node generates and combine them into a single work item. It will also prevent any descendent tops from cooking until all the work items above have completed cooking.



03 Add a **Geometry Import** TOP. Name it *building_merge*. Set **Generate When** to **Each Upstream Item is Cooked** and set **Merge Operation** to **Import and Merge All Geometry**.

Save your scene file then select the *building_merge* node and press **Shift-G** to cook it. When the network finished cooking, click on the work item on the final *building_merge* node. You will see the whole city in the viewport.

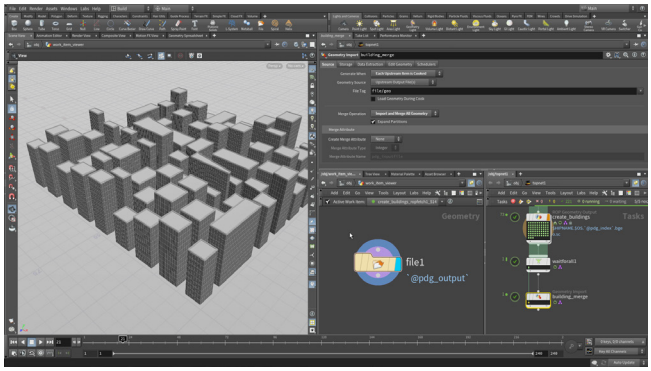


04 Go back to the **Attribute Create TOP** and edit the *height_variation* attribute to the following:

- **Value** to `rand(@pdg_index) * 15`

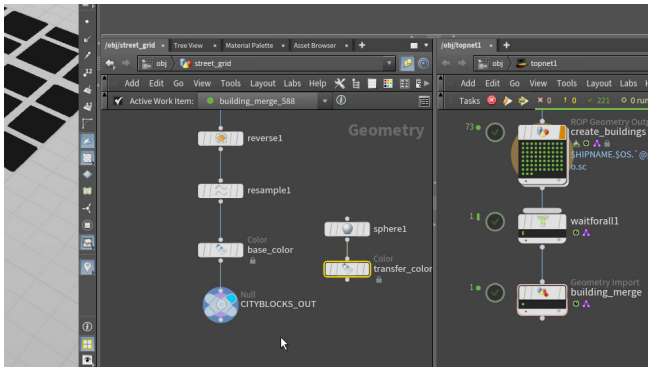
This will make the buildings a bit shorter overall.

RMB-click on the *attributecreate* TOP and choose **Dirty This Node**. This will clear out all the work items going down the chain.



05

Save your scene file. Now select the *building_merge* TOP node and press **Shift-G** to cook it. When the network finishes cooking, click on the work item on the final *building_merge* node to see the change.



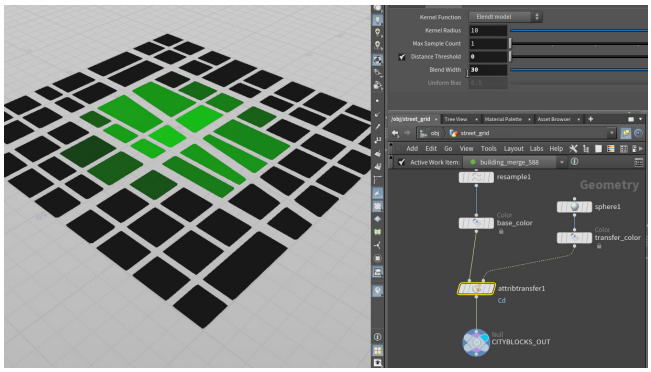
06

Navigate to the *street_grid* object and dive into it. Create a **color** node and wire it into the chain after the *resample* node. Set the **Class** to **Primitive** and the **Color** to **black (0, 0, 0)**. Rename this node *base_color*.

Create a sphere in the *street_grid* network and place it off to the side and set:

- **Primitive Type** to **Polygon**
- **Uniform Scale** to **5**

Add a Color node to the output of the *sphere* and set **Class** to **Primitive** and **Color** to **green (0, 1, 0)**. Name this node *transfer_color*.



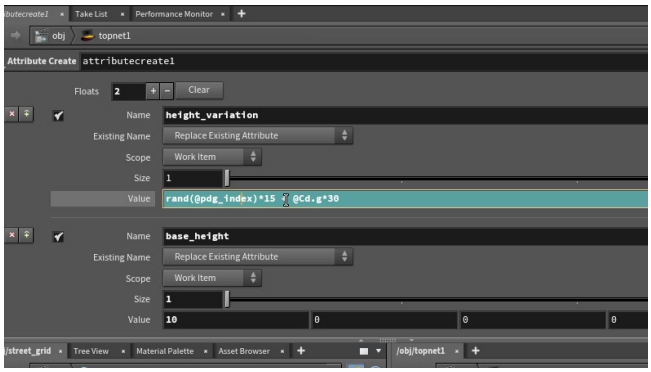
07

Add an **Attribute Transfer** node after the *base_color* node. Then connect the *transfer_color* node into its second input.

Uncheck the **Points** checkbox, and set the **Primitives** field to **Cd**. Then in the **Conditions** tab, set:

- **Distance Threshold** down to **0**
- **Blend Width** to about **30**.

With your cursor over the Scene view, press **Spacebar-Y** to google to **Hide Other Objects**. You can now see the green color gradually permeate the tiles as they approach the center of the sphere.



08

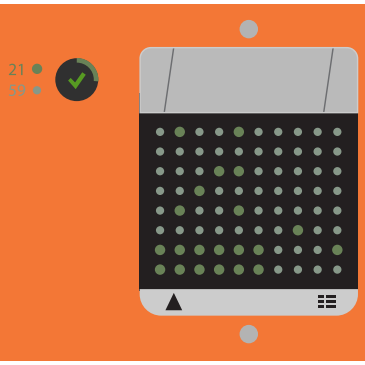
Go back to the TOP network and select the *attributecreate* node and edit the *height_variation* attribute to the following:

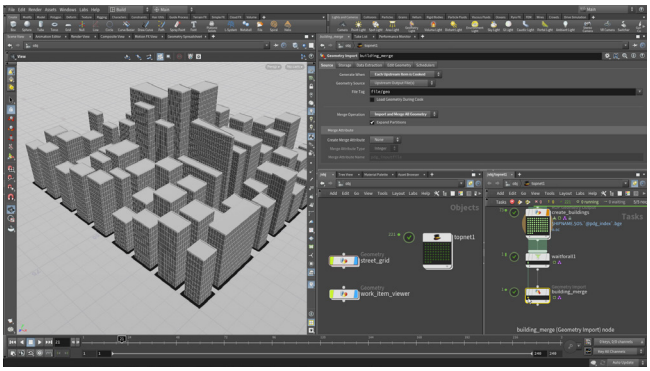
- **Value** to `rand(@pdg_index) * 15 + @Cd.g*30`

This will add height to those buildings in the city core. The value of 30 will be the maximum height added. You can change this value if you want taller or shorter buildings.

DIRTY & CLEAN WORK ITEMS

One of the main purposes of the PDG technology is the dependencies that arise in a complex system. As you make changes to the downtown core, you will see that some nodes become dirty and have to be recooked while others are fine the way they are. The way TOPs handles these dependencies is one of its strengths.

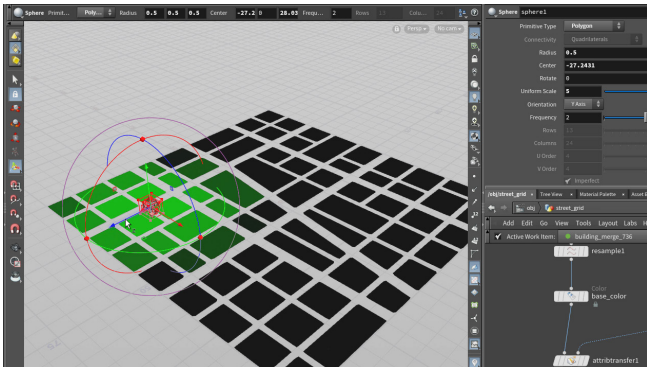




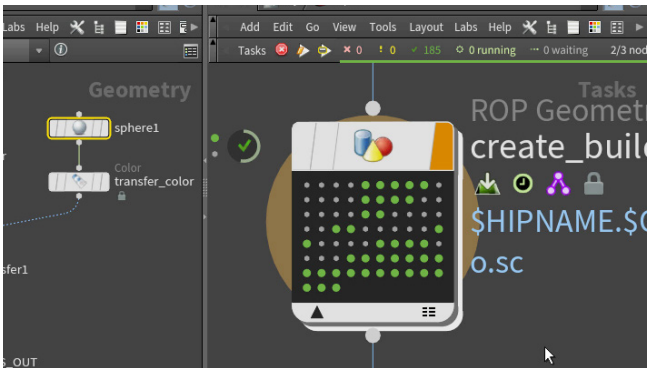
09 **RMB-click** on the *building_merge* node and from the menu select **Dirty This Node**. You could also select it and hit **Shift-D** to dirty it.

Select the *building_merge* node and press **Shift-G** to cook it. You will be prompted to Save. When it is ready, click on the work item on the final *building_merge* node.

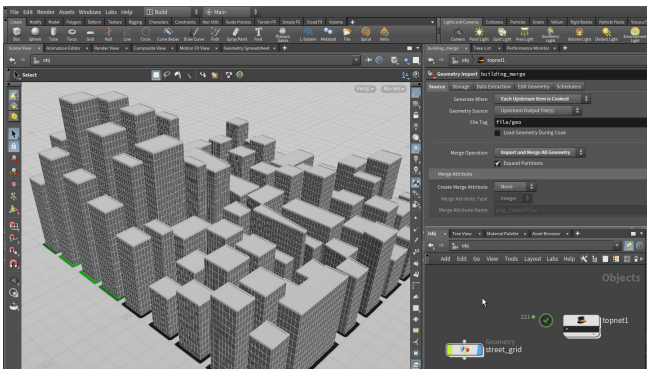
In the other Network view, navigate back up to the object level. You will see the city updated with the city core in the viewport.



10 Navigate to the *street_grid* object and with the *CITYBLOCK_OUT* node displayed, you can select the *sphere* node and use the **handle** tool to move it around to see it affect a different part of the city grid.



11 Go back to the TOP net and **RMB-click** on the *create_buildings* node and select **Generate Node.** You will see that some of the work items have been "dirtied" automatically because of the change at the geometry level. The other work items are still considered clean and will not need to be recooked.

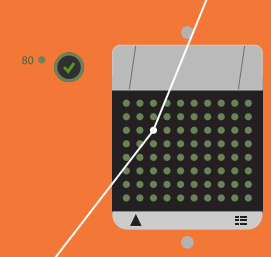


12 Now when you **RMB-click** on the *building_merge* node and select **Cook Selected Node**, only the dirty work items will be updated. Click on the work item to see the new city core. This is one of the ways that the dependency graph in TOPs works efficiently as you develop your workflow.

UPDATED DEPENDENCIES

When you select on a work item, you get lines that show how that work item is connected to other work items in the system.

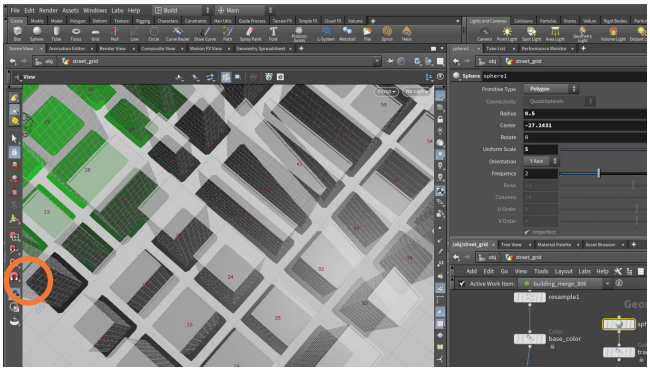
This is a mapping of the dependencies and this helps your graph work efficiently because only dirty work items will be processed unless you explicitly dirty all the work items on a node with **Shift-D**.



PART SIX

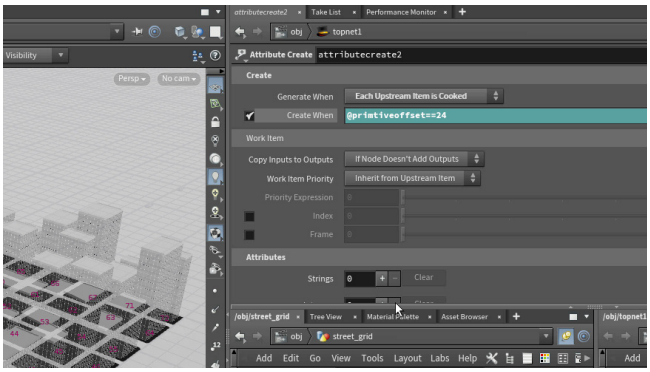
Isolate a Building

Up until now the building heights are determined by the randomness of the height_variation. If there is one building that you want to fix to a certain height, a second attribute create node can be used to choose a primitive and set specific values. This makes it easier to art direct a system when you want to override the randomness.



01 Press **Spacebar-Y** to google to **Ghost Other Objects**. Go back to the *street_grid* object and turn on **Display Primitive Numbers** in the **Display Options**. You can see the block numbers. You can identify the buildings and decide if you want to set one of them explicitly. Here **block 24** is the chosen block.

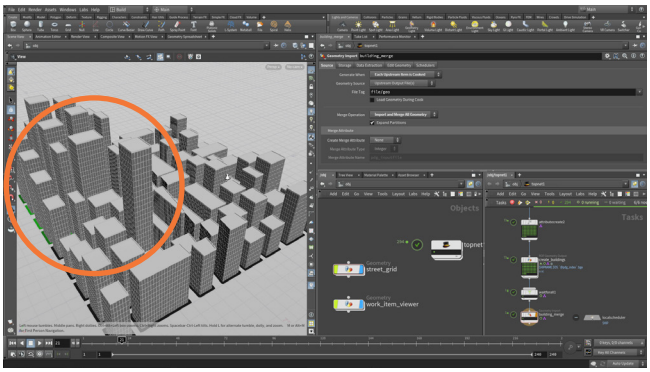
You are now going to set specific parameters for this city block to get the exact height you want instead of relying on the randomness of our current setup. This will give you some artistic control within the TOP network instead of relying on the randomness to determine everything.



02 Now **Alt-click**-drag on the *attributecreate* TOP node to make a copy of it. Don't wire it in yet. Turn on the checkbox next to **Create When** and add the following expression:

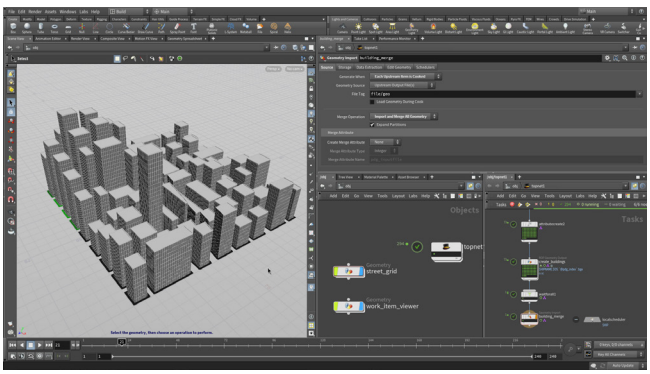
@primitiveoffset==24

RMB-click on *height_variation* **value** parameter and choose **Delete Channels** then set its value to **0** and then set *base_height* value to **50**. This will explicitly make the building on **block 24** exactly 50 units high.



03 Insert this node in between the first *attributecreate* and the *create_buildings* node. **Save** your work then recook the *building_merge* node.

Click on the work item at index 24 on the *create_buildings* node to see the building on block 24 at 50 units.



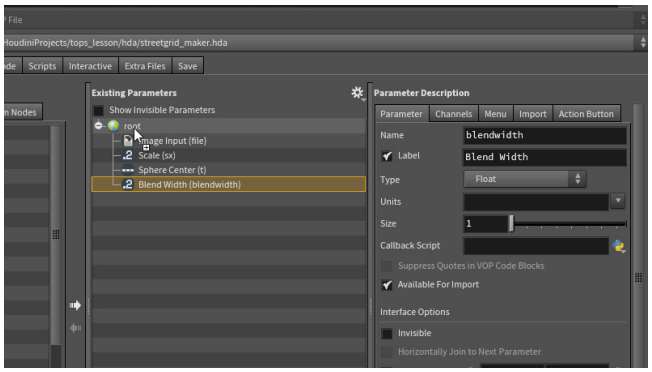
04 Click on the work item on the *building_merge* node to see the new cityscape with the new building rising up. Now anytime you recook the network, the building on block 24 will not update but will remain set specifically.

If you want to change that building, you can use the second attribute create node to explicitly set its height.

PART SEVEN

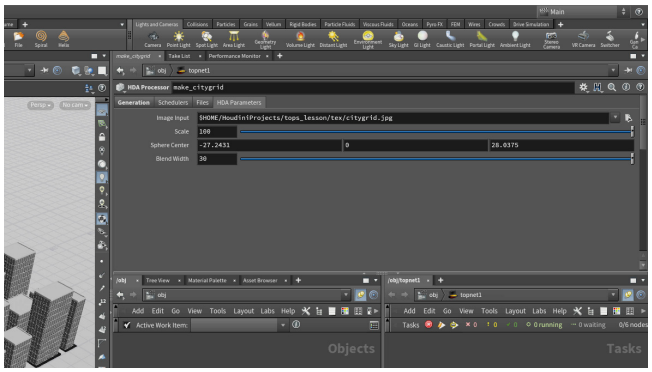
Wedge the City Core Location

In order to wedge the position of the sphere within the street_grid object, the network will need to be turned into a Houdini Digital Asset [HDA] and run through an HDA Processor TOP node. The ability to load these assets and add them to the system lets you create a complex system out of discreet tools that can be updated and adapted as needed.



01 In the *street_grid* network, select all the nodes except *CITYBLOCKS_OUT*. From the **Assets** menu, choose **New Digital Asset from Selection.** Name the asset *streetgrid_maker*, label it *Street Grid Maker*, and save it to the `$HIP/hda/` directory.

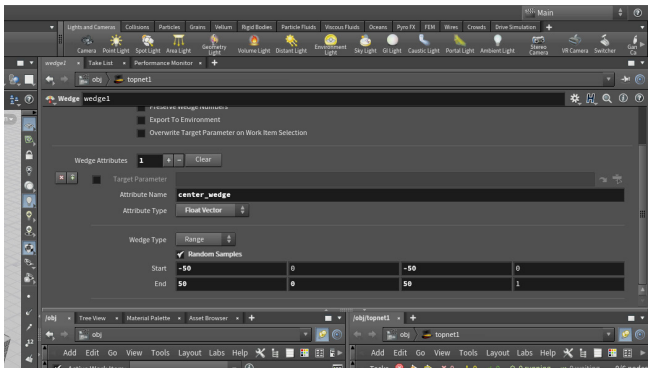
In the **Operator Type Properties** panel, click on the **Parameters** tab. Dive into the asset. Drag **Image Input** from the *trace* node to the parameter list. Drag **Scale X** to **Scale Y** and choose **Relative Channel Reference**. Now drag **Scale X** to the parameter list and name it *Scale*. Drag **Center** from the *sphere* node and name it *Sphere Center*. Drag **Blend Width** from the *attributetransfer* node then click **Accept**.



02 Go back to the TOP network. Create an **HDA Processor TOP** and wire it into the **Geometry Import** node. On the *geometryimport* node, change **Generate When** to **Each Upstream Item is Cooked** and **Geometry Source** to **Upstream Output File**.

Select the **HDA Processor** node and set **HDA File** to *$HIP/hda/streetgrid_maker.hda*. Click on the **HDA Parameters** tab to see the **Image Input**, **Center, Scale** and **Blend Width** parameters listed.

Rename this TOP *make_citygrid* then press **shift-V** to *Dirty and Cook* the node to make sure that it is still generating the city grid the way it was before.



03 Create a **Wedge** TOP node and wire it into the *make_citygrid*. Set the following:

- **Wedge Count** to **4**

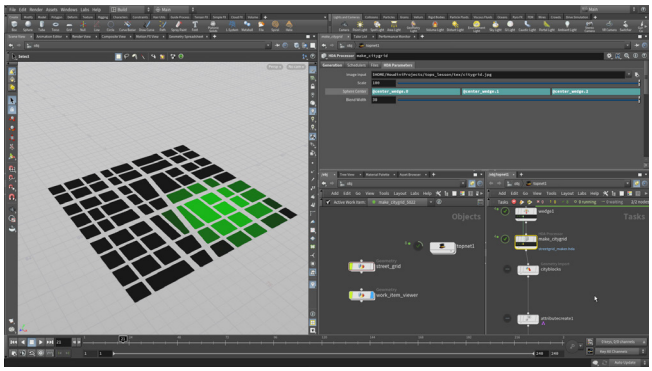
Click the **plus sign** next to **Wedge Attributes** and set the **Attribute Name** of the first one to *center_wedge* then set:

- **Type** to **Float Vector**,
- **Start range** to **-50, 0, -50, 0**
- **End range** to **50, 0, 50, 0**
- Turn **ON** the **Random Samples**

WEDGING

Wedging is an idea that comes from photography where a shot is taken using a variety of settings so that the best one can be chosen later in the lab. Here the idea is the same except you will use random values to influence the system and get four unique results that can be compared.

Later you will render out a mosaic with all the options labelled with the random wedge values displayed.



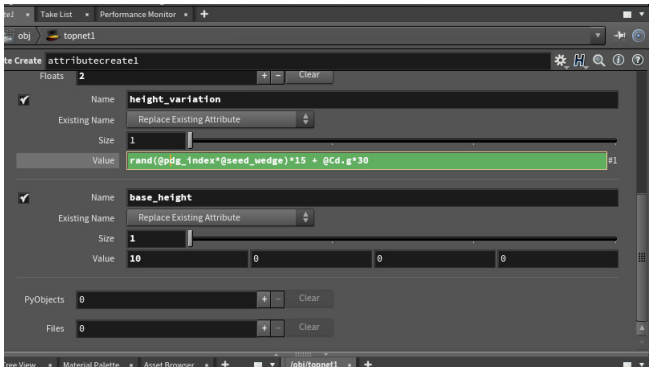
04

Select the **HDA Processor** node and on the **HDA Parameters** tab, set the **Center** parameter to:

```
@center_wedge.0, @center_wedge.1, @center_wedge.2
```

Select the *make_citygrid* HDA processor node and press **shift-V** to *Dirty and Cook* the node - now there are four citygrids being created.

Hide the **street_grid** object. Click on the work items to visualize the different grids. Each of them show a different position for the green color which is where the city core is located.



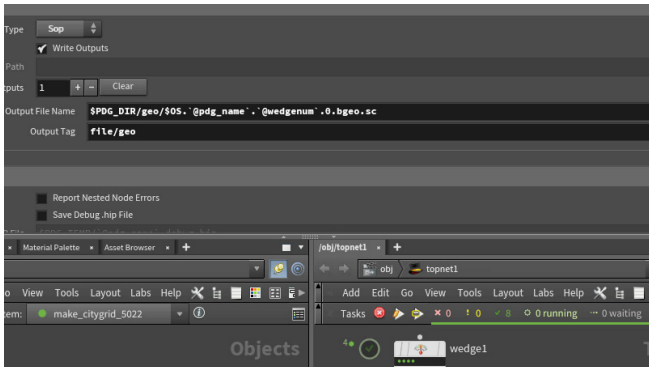
05

Now add some wedging to the height variation so that you get four different looks. Select the *wedge* TOP and click the **plus sign** next to **Wedge Attributes** and set the **Attribute Name** of the second one to *seed_wedge*, leave **Type** set to **Float**, and set its **Start/End** to **0, 1000**.

Turn **ON** the **Random Samples** parameter.

On the *first attributecreate* TOP, change the *height_variation* attribute's **Value** to:

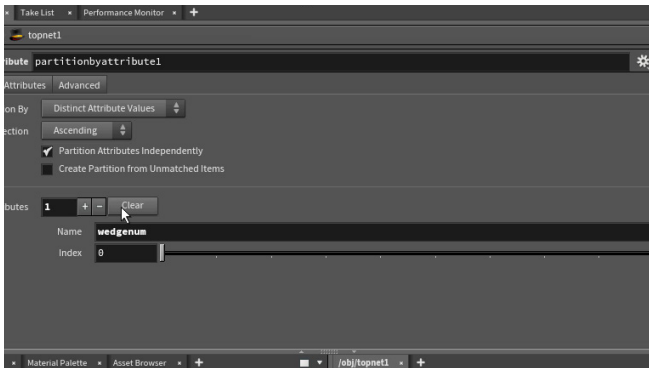
```
rand(@pdg_index*@seed_wedge) * 15 + @Cd.g*30
```



06

You will have to update the output file names for all the TOPS that create an output file to include the wedgenumber, so that the files can be differentiated for the different wedges. Include a `.`@wedgenum`` in the output file parameters for the following TOP nodes just before `.bgeo`:

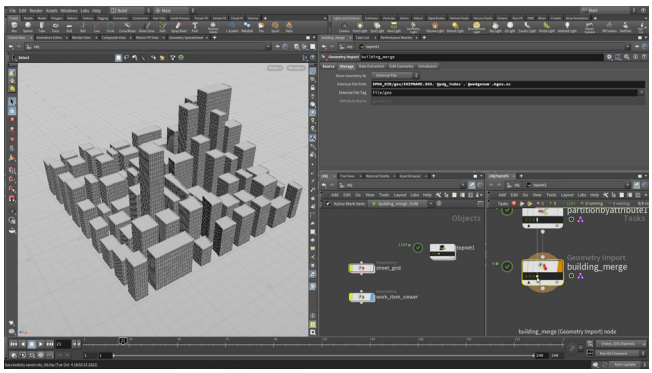
- *make_citygrid* HDA Processor
- *geometryimport* Geometry import
- *create_buildings* ROP Geometry
- *building_merge* Geometry Import



07

Replace the **Wait for All** node with a **Partition by Attribute** TOP node. Make sure that **Partition By** is set to **Distinct Attribute Values**.

Click the **plus sign** next to **Attributes** and set the **Name** to *wedgenum*. This will create a partition for each wedge.



08

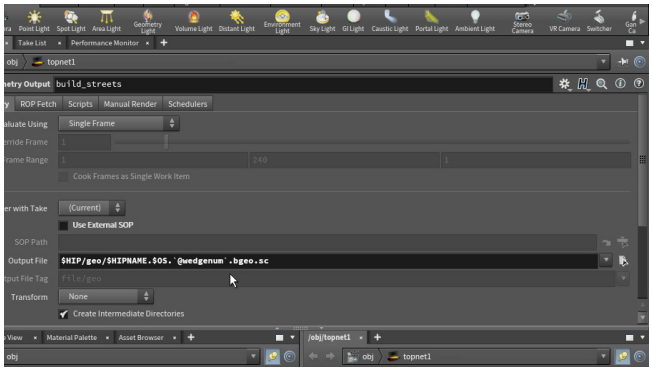
Save your work then cook the *building_merge* TOP node. This will create the four city maps which you can visualize by clicking on the work items on the *building_merge* node.

Since this is using the local scheduler this will take a little while longer to process. This is where using a scheduler that distributes your tasks to a compute farm can speed things up considerably.

PART EIGHT

Create Geometry for the Streets

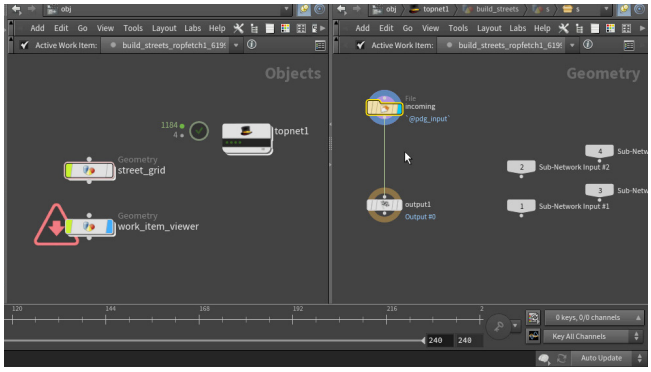
To create context for the buildings, you will now create city blocks and streets to use in the final rendering. While the four wedges currently all use the same map, you will set up this geometry in TOPs to allow for the use of different maps down the line. It is always a good idea to build in flexibility to create the most robust system.



01 In the TOP network, add a **ROP Geometry Output** node and branch it off from the *make_citygrid* HDA processor. Rename this new node *build_streets*.

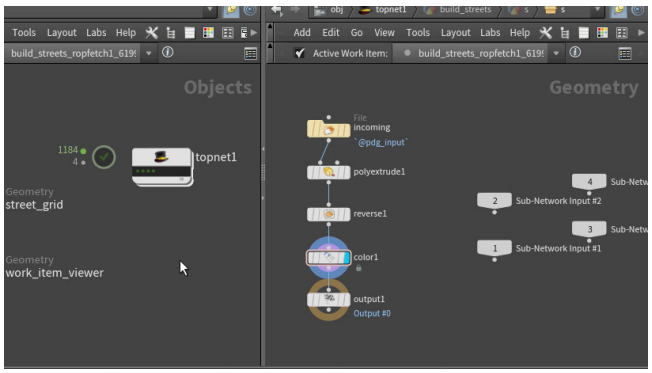
Turn **OFF** the **Use External SOP** option and in the **Output File** add `.`@wedgenum`` to the expression in place of $F just before `.bgeo`.

On the **ROP Fetch** tab, set **Cache Mode** to **Write Files**.



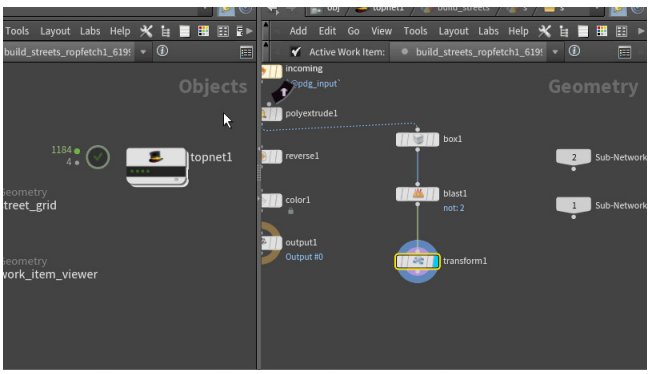
02 **RMB-click** on this node and choose **Generate Node** to get the four work items. Click on one of them to highlight the work item and then double click on the node to dive into it.

This will bring you to the geometry level where you will create the streets.



03 Add a **PolyExtrude** node between *incoming* and *output* nodes. Set **Distance** to **-0.05**. Turn **OFF** the **Output Front** checkbox and turn **ON** the **Output Back** option.

Add a **Reverse** node after to fix the normals and then add a **Color** node to turn all the city blocks white.

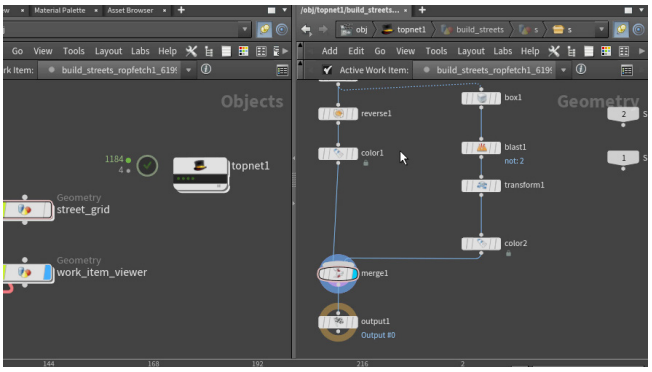


04 Add a **Box** node into the network off to the side. Wire *polyextrude* into the *box* node to match the bounding box to the city grid geometry.

Add a **Blast** node after the *box* and set **Group** to **2** and turn **ON** the **Delete Non Selected** option.

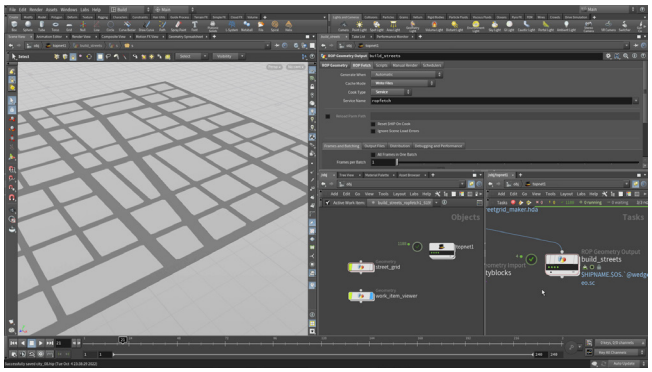
Add a **Transform** node after the *blast* and set:

- **Translate Y** to **-0.05**
- **Scale X** to **1.05**
- **Scale Z** to **1.05**



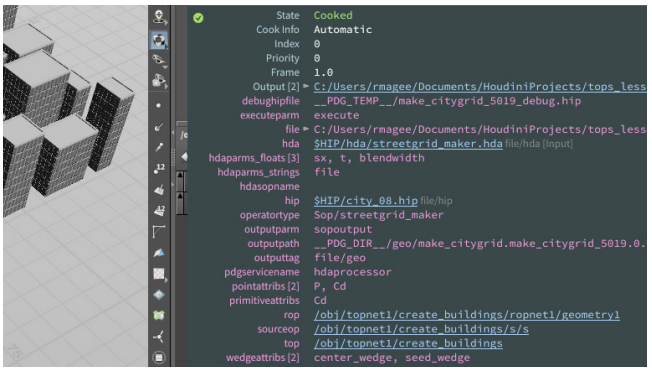
05 Add a **Color** node with a medium grey (0.33, 0.33, 0.33).

Now create a Merge node and connect it to the two *color* nodes. Make sure the **display flag** is set to output to see the street geometry in place.



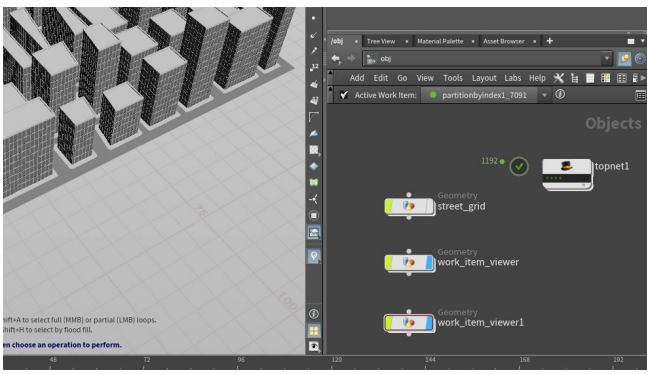
06 **Save** the Scene File. Go back to the TOP Network and Cook the node, you can click on the work items to see the grid.

At the moment all four of them are the same. That will be different later when you introduce more city grids into the equation.



07 Add a **Partition by Index** to the end of the chain. Feed the *building_merge* node into the first input and the **build_streets** into the second.

Cook the new node. When you are finished, click on the work items and you will see that only the buildings are being displayed. If you **middle click** on a work item you will see that there are two output files being created by the Partition. You need to display the new one to see it in the viewport.



08 Go to the Object level. **Alt drag** on the *work_item_viewer* to create a second one called *work_item_viewer1*. Dive into the geometry level and set Geometry File to:

```
`@pdg_output.1`
```

Go back up one level to the object level then in the TOP network select one of the work items on the *partitionbyindex* node. Now you will see both of the outputs being displayed in the viewport. This is important for the next step when you will render out images of the city.

ANOTHER PDG OUTPUT

Earlier you used **pdg_output** to create the **work_item_viewer** node to help you visualize what is coming out of each node. Now that you have this partition in place, there are actually two outputs which you can see by **Ctrl-MMB** clicking on a work item. This second output therefore needs its own viewer to be properly displayed.

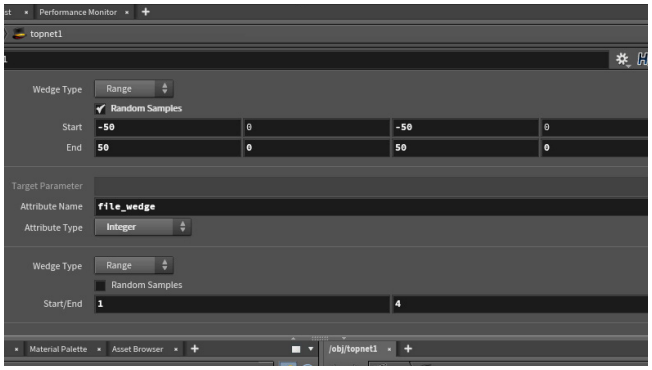
Output [2]

C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/city_01.building_merge.1.1.bgeo.sc file/geo
C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/city_01.build_streets.1.bgeo.sc file/geo

PART NINE

Wedge Four City Maps

Let's add one extra variable into the wedging. You are going to access four different maps and create one rendering for each one of them. Each of the different black and while images will be traced and used to feed the system. This shows another way to add more content into the pipeline.

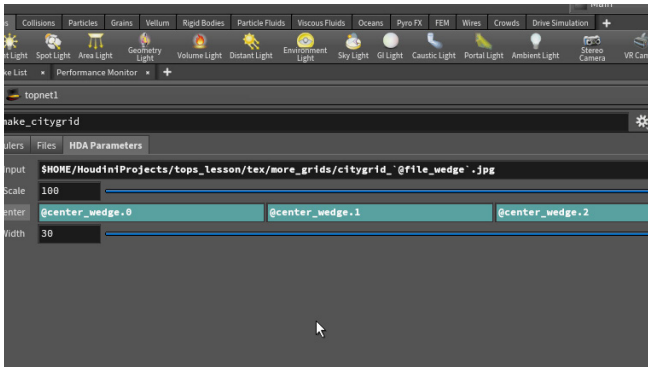


01 Go back to the *wedge* TOP node. Click the **plus sign** next to **Wedge Attributes** to add a new wedge parameter.

Set the following:

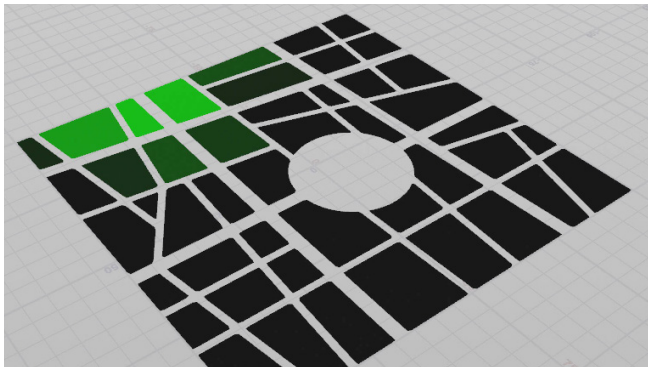
- **Attribute Name** to *file_wedge*
- **Attribute Type** to **Integer**
- **Start/End** to **1,4**

This will create integer values of 1, 2, 3 and 4 which will be used to find the different city maps.

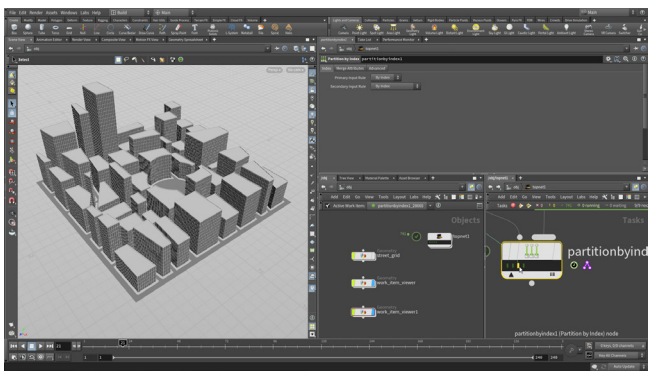


02 On the *streetgrid_maker* HDA processor node, click on the **File** button next to **Image Input** and navigate to the *more_grids* folder. Select the file from there then in the Parameter pane, change `$F` in the expression to `` `@file_wedge` ``

There are four files in this folder and the wedge attribute will run through them all in order when you cook the network.



03 Dirty and Cook (**Shift-V**) the *streetgrid_maker* HDA processor node. You will see the four different maps being used and the city core being moved around like before.



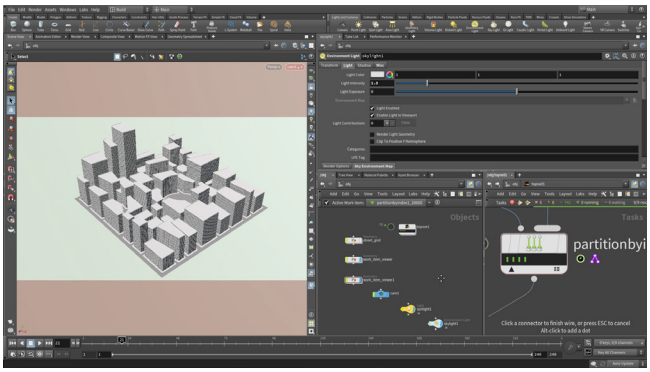
04 **Bypass** the second *attributecreate* TOP node. This was designed for a static map and won't work for four different maps.

Cook the Final partition node (**Shift-G**) and view the four city plans turned into cities.

PART TEN

Render a Mosaic

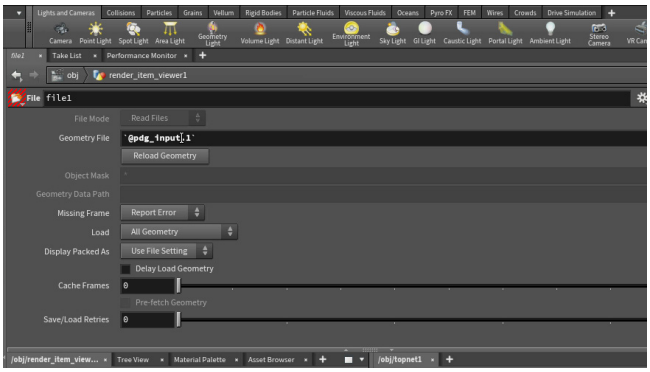
Now you are going to add a camera and a skylight then render out the cities. The resulting images will then be combined into a single mosaic with the wedge attributes on display so that creative decisions can be made based on the results. The mosaic node will use the Image Magick application which needs to be installed on your computer for these steps to work.



01

Tumble your view until you are looking down at the city from an angle. From the **camera** menu, select **New Camera**. Check the four city grids to make sure they all fit in the camera. Adjust the camera view if necessary. You will use this to render out the mosaic.

Add a **Skylight**. Set the Environment Light **Intensity** to **1.3**.



02

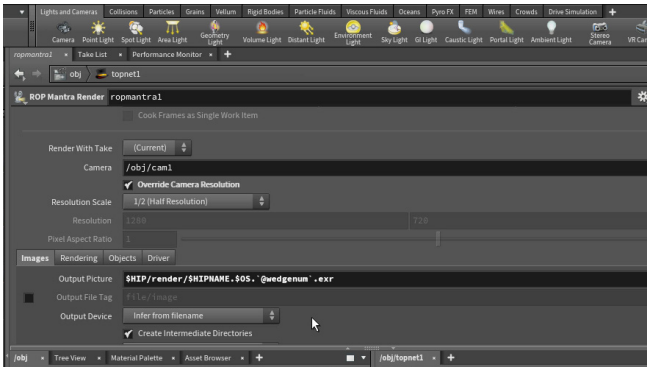
At the object level , **Alt-drag** on *work_item_viewer* to make a copy. **Rename** the new node *render_item_viewer*. Dive into this node and change the **Geometry File** parameter to:

```
`@pdg_input`
```

Repeat these steps to create *render_item_viewer1* with its **Geometry File** parameter set to:

```
`@pdg_input.1`
```

Set the display flag on these new *render_item_viewer* nodes and **turn off** the display on the two *work_item_viewer* nodes.



03

Append a **ROP Mantra Render** top after the *partitionbyindex*. On the **ROP Fetch** tab, set the **Cache Mode** to **Write Files**.

Make sure the camera parameter is set to the camera you made. Turn **On** the **Override Camera Resolution** option and set it to ½. Change the **Output Picture** parm to:

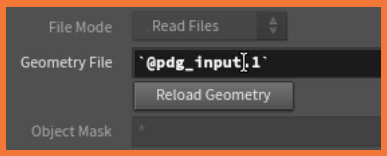
```
$HIP/render/city.$OS.`@wedgenum`.exr
```

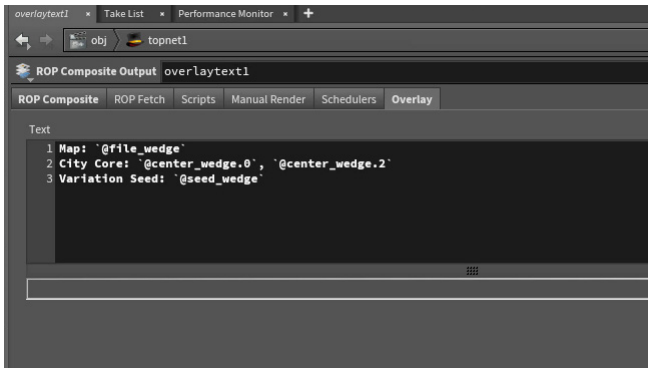
Note: If you are running this tutorial using Houdini Apprentice then you should use `.pic` files instead of `.exr` to avoid the Apprentice watermarks.

PDG INPUT

While PDG OUTPUT lets you take the result of a node and display it, PDG INPUT will take the result of the node before and display that instead.

This way the Mantra node can use the results from the preceding node as the input for the geometry and then render that geometry. These two options are similar but understanding the difference is important.





04

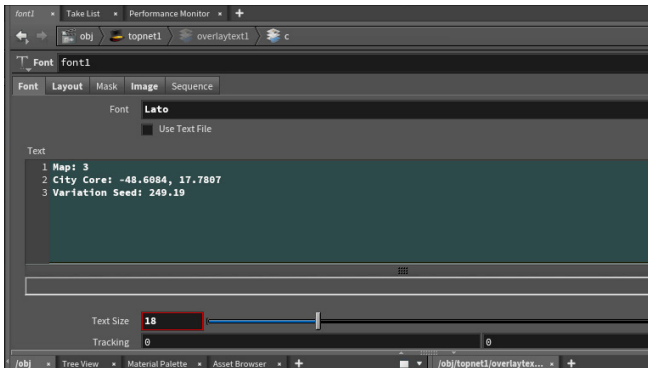
Append an **Overlay Text** top and set **Output Picture** to:

`$HIP/render/$HIPNAME.$OS.`@wedgenum`.exr`

On the **ROP Fetch** tab, set **Cache Mode** to **Write Files.**

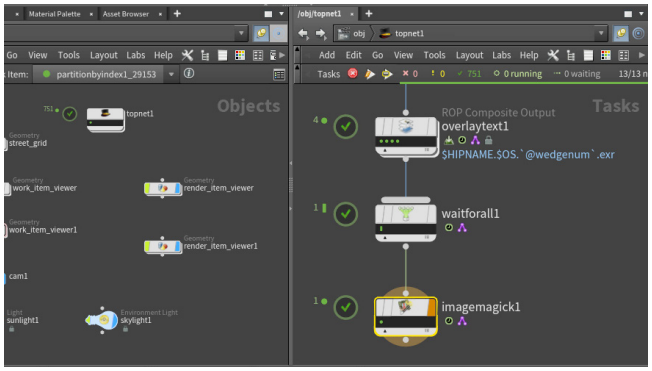
In the **Overlay** tab type:

```
Map: `@file_wedge`
City Core: `@center_wedge.0`, `@center_wedge.2`
Variation Seed: `@seed_wedge`
```



05

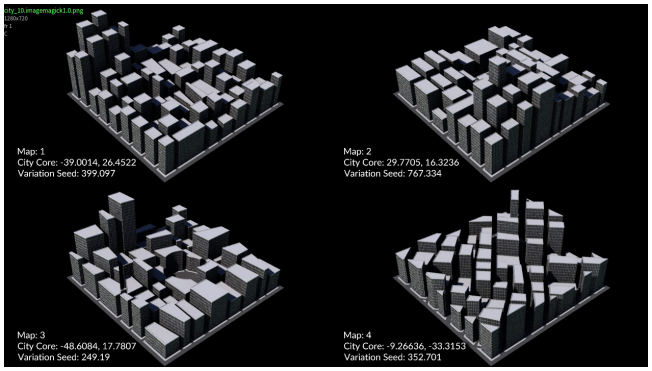
Double-click on the *Overlaytext* node to dive into it. Select the *Font* node and change the **Text Size** to **18**. This will create smaller text and leave more room to see the city.



06

Go back up and add a **Wait for All** partition node. This will bring together all of the elements. Now add an **ImageMagick** node.

Cook this node to process the TOP network. This will create four renderings and then overlay the text and combine them into a mosaic.



07

When you are finished, **RMB click** on the work item and select **View Task Output.** This will show you the final image in the **Mplay** image viewer.

This contact sheet lets you review the different wedges and decide which one you want to move forward with. The parameters displayed on the images are the ones you would need to lock down to get the city you want.

PDG FOR MORE THAN HOUDINI

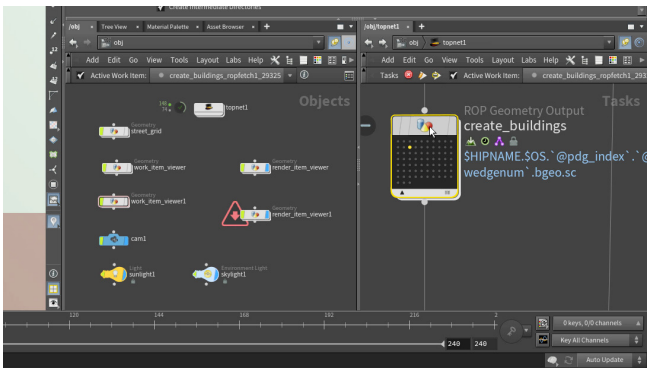
With PDG's ability to work with Python, you can use it for tasks that are not directly connected to Houdini. It offers a visual tool for organizing your pipeline tools with TOP nodes organizing your workflow.

These networks can be processed through Houdini and sent to your chosen scheduler. This will allow you to send your work to a compute cloud for faster processing.

PART ELEVEN

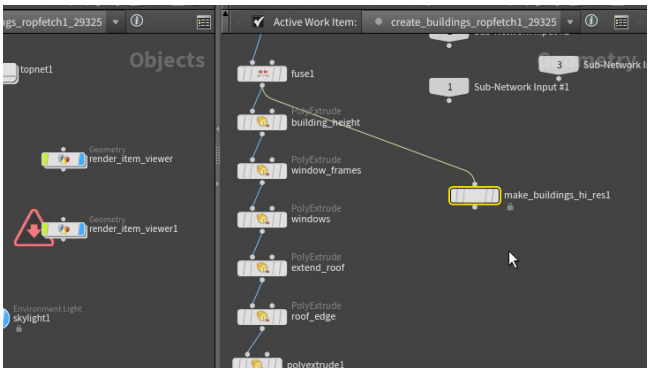
Scale Up to Create More Content

In the beginning you pointed out that this city building example could probably be created in SOPs without too much difficulty. The issue is that as the system gets more complex there is a bottleneck since all the processing happens within a single network. With TOPs, you can distribute work items to a compute farm and as you scale up things don't have to slow down.



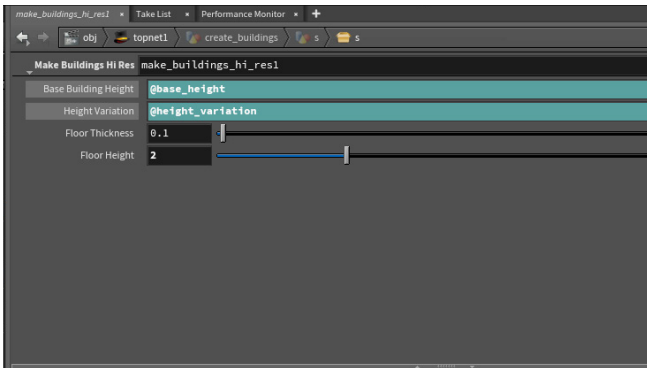
01 On the wedge TOP, set the Wedge Count to 1. This will give you only one iteration for now while you scale up the city using a new map.

Select *create_buildings* ROP Geometry TOP node and **RMB-click** and choose **Generate Node**. Now you are going to dive in and use a digital asset to create a different type of building with floors and windows and columns.



02 **Double-click** to dive down. From the **Assets** menu, select **Install Asset Library**. Navigate to the *hda* directory and select the *make_buildings2.hda.* Click **Accept** then click **Install** (not Install and Create) Press **tab > Make Buildings Hi Res** to place the node into the graph.

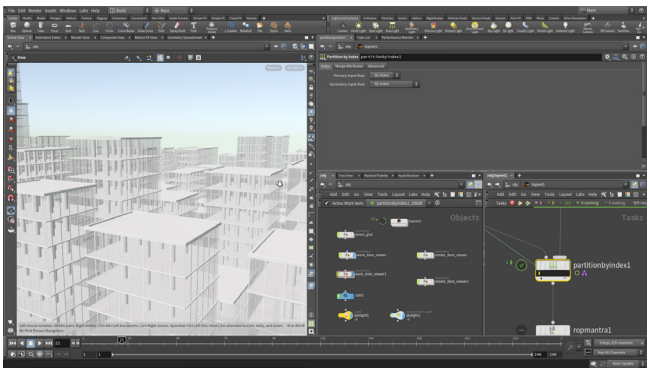
Wire the *fuse* node into the new asset.



03 On the *make_buildings_hi_res* digital asset, set the following:

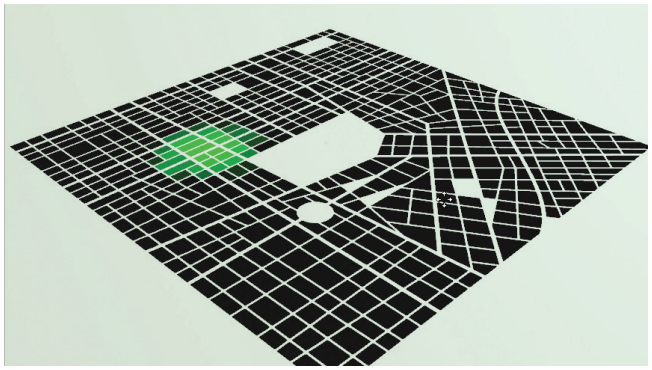
- **Floor Height** to **2**.
- **Base Building Height** to `@base_height`
- **Height Variation** to `@height_variation`

Add a *switch* node between the *color* node and the *output* node then wire the *make_buildings_hi_res* node into the second input and set **Select Input** to 1.



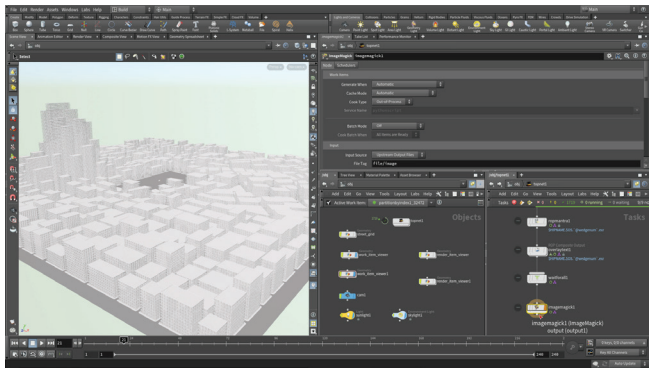
04 Select the **Partition by Index** node and **Cook** the network to see the new buildings. Now they have floor slabs, columns and windows.

A lot more geometry has been produced and you are starting to see that increasing the number of available work items using a compute farm would be a good idea. You could do this using one of the other scheduler nodes.



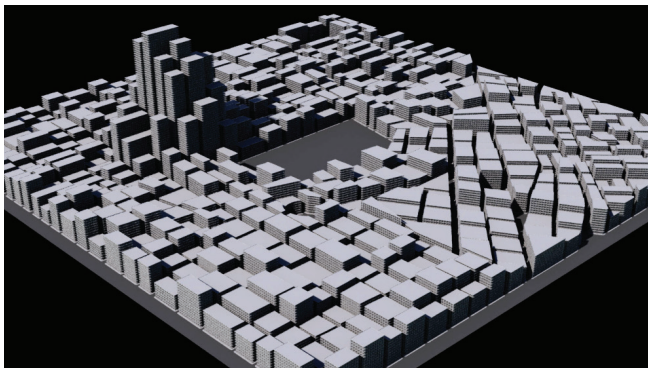
05 You could also make the city bigger using a different city grid image. You would need to increase the **Uniform Scale** on the *streetgrid_maker* HDA processor to **500**, and change the **Image input** to the *citygrid_large* map. You may also want to increase the **Blend Width** to create more falloff in the city core.

On the Wedge node, you could set the *center_wedge* attribute's to something like -110, 0, 100.



06 **Dirty and Cook** the *streetgrid_maker* HDA processor node. You will see the new city map being generated. When this is finished, you will have a lot more city blocks, compared to the original map and a lot more buildings are being generated.

Going to a compute farm would definitely make sense at this point because a single computer is not taking advantage of the parallel processing capabilities of TOPS.



07 **Recook** the final node to render out an image of the city. You can see how the simple system you built has the potential to grow to whatever size you need.

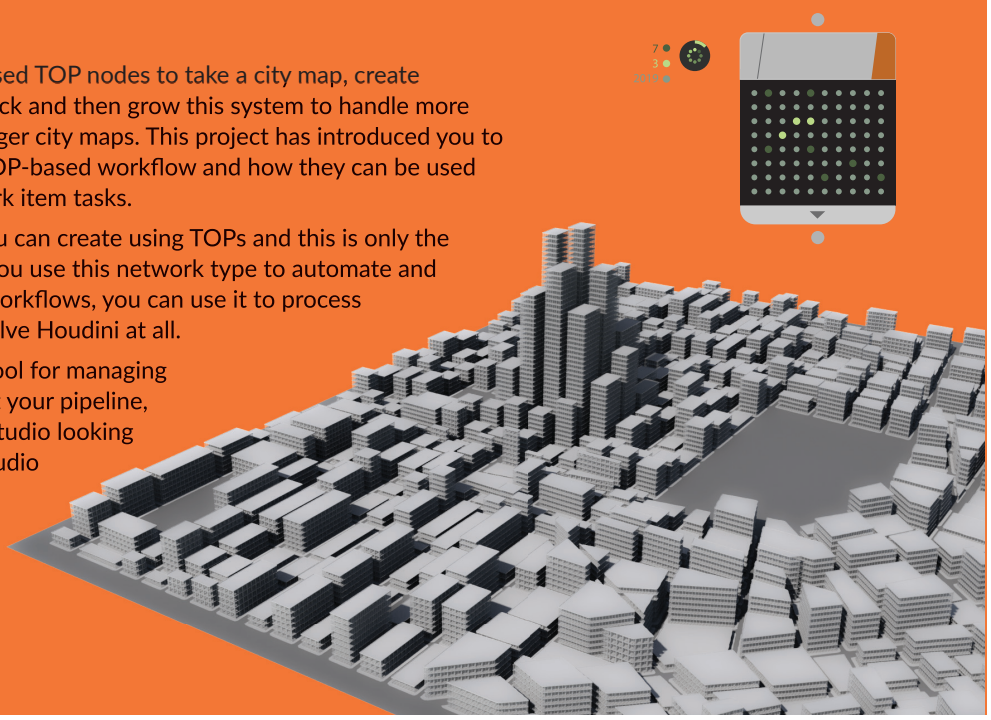
Save your scene.

CONCLUSION

In this lesson, you have used TOP nodes to take a city map, create buildings for each city block and then grow this system to handle more complex buildings and larger city maps. This project has introduced you to typical nodes used in a TOP-based workflow and how they can be used to create and process work item tasks.

There are many things you can create using TOPs and this is only the beginning. Not only can you use this network type to automate and process typical Houdini workflows, you can use it to process workflows that don't involve Houdini at all.

This will make it a great tool for managing dependencies throughout your pipeline, whether you are a small studio looking for efficiencies or a big studio managing lots of data.



SCENE VIEW SHORTCUTS

TOOLS

Select	S
Move	T
Rotate	R
Scale	E
Pose	Ctrl-R
Handle	Enter
View	Esc
Tool Menu	Tab
Custom Radial Menu	C
Repeat Last Tool	G

VIEW

Tumble	Space/Alt + LMB
Track	Space/Alt + MMB
Dolly	Space/Alt + RMB
Home Grid	Space + H
Home All	Space + A
Home Selected	Space + G

VIEW RADIAL MENU

Selection Tools	V →
Selection Options	V ↑
Viewport	V ←
Shading	V ↓

SELECTION MODES

Objects	1
Points	2
Edges	3
Primitives (Faces)	4
Vertices	5
Select Groups/Connected Geometry	9
Toggle Objects/Geometry	F8

SELECTING

Select	LMB
Add to Selection	Shift + LMB
Remove from Selection	Ctrl + LMB
Toggle Selection	Ctrl + Shift + LMB
Select All	N
Select Nothing	Shift-N

SNAPPING RADIAL MENU

Grid Snap	X →
Primitive (Curve) Snap	X ↑
Point Snap	X ←
Multi-Snapping Snap	X ↓

VIEWPORTS

Expand Viewport	Space + b
Select Viewport	Space + n
Perspective View	Space + 1
Top View	Space + 2
Front View	Space + 3
Right View	Space + 4
UV View	Space + 5
Toggle Wireframe/Shaded	W
Display Options	D

VIEWPORT LAYOUT

Single View	Ctrl + 1
Four Views	Ctrl + 2
Two Views Stacked	Ctrl + 3
Two Views Side by Side	Ctrl + 4
Three Views Split Bottom	Ctrl + 5
Three Views Split Left	Ctrl + 6
Four Views Split Bottom	Ctrl + 7
Four Views Split Left	Ctrl + 8

FINDING THINGS

Dashbox	Ctrl + D

NETWORK VIEW SHORTCUTS

VIEW

Pan	Space + LMB or MMB
Zoom	Space + RMB or Scroll Wheel
Show all Nodes	H
Show Selected Nodes	G

CREATE

Node Menu	Tab
Add File Node	=
Create Subnet	Shift + C
Add Background Image	Shift - I

NOTES AND NETWORK BOXES

Add Network Box	Shift + O
Add Sticky	Shift - P
Minimize Selected Notes/Boxes	Shift - J
Expand Selected Notes/Boxes	Shift - K
Shrink box to fit contents	Shift - M

WIRING

Connect Nodes	LMB on Connector
Connect Multiple Nodes	J drag over nodes
Insert Node	RMB on Connector
Branch	MMB on Connector
Connector List	Alt + MMB on Node
Cut Wire	Y drag across wire
Disconnect from Wires	Shake Node

DOTS

Add Dot	Alt + LMB on wire
Pin/Unpin Dot	Alt + LMB on dot

TOOLS

Toggle Parameter Pane	P
Toggle Tree View	Shift + W
Toggle Network Overview	O
Toggle Color Palette	C
oggle Shape Palette	S

CLICKS AND DRAGS

Select	LMB
Add to Selection	Shift + LMB
Remove from Selection	Ctrl + LMB
Start Wiring from Node	Alt + LMB
Select Node + Inputs	Alt + Shift + LMB
Select Node + Output	Alt + Ctrl + LMB
Select Inputs + Outputs	Alt + Shift + Ctrl + LMB
Move Node	LMB-Drag
Move Node + Inputs	Shift + LMB-Drag
Move Node + Outputs	Ctrl + LMB-Drag
Copy Selected Nodes	Alt + LMB-Drag
Copy Node + Inputs	Alt + Shift + LMB-Drag
Copy Node + Output	Alt + Ctrl + LMB-Drag
Reference Copy	Alt + Shift + Ctrl + LMB-Drag

NAVIGATION

Enter a Node	Double-click or Enter
Go up a level	U
Radial Menu	N
Create a Quickmark	Ctrl + <# 1-5>
Go to a Quickmark	Shift + <# 1-5>
Go to Previous View	` (Backtick)
Select the Node Upstream	PgUp
Select the Node Downstream	PgDn
Select Previous Sibling	Shift + PgUp
Select Next Sibling	Shift + PgDn

ORGANIZE NODES

Layout all	L
Align	A + LMB-Drag Down/Across

DISPLAY FLAGS | SOP LEVEL

Render	T + LMB
Display	R + LMB
Template	E + LMB
Footprint	W + LMB
Bypass	G or B + LMB