SideFX®

Houdini™ FOUNDATIONS

FOR FILM, TV & GAMEDEV

VERSION 19.5 JAPANESE EDITION

Houdini™
FOUNDATIONS

FOR FILM, TV & GAMEDEV

VERSION 19.5 JAPANESE EDITION

ROBERT MAGEE

SideFX®

HOUDINI FOUNDATIONS

Author: Robert Magee
Research: Michael Buckley, Marcia Utama
Cover Art: Attila Torok
Japanese translation: B-Sprout Inc.

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

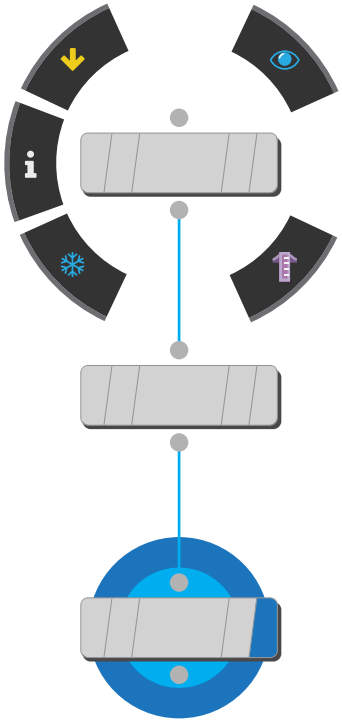
Written for the features found in Houdini 19.5
Document Version 2.0 | December 2022

HOUDINI FOUNDATIONS

Contents

HOUDINI FOUNDATIONS

概要



映画、テレビ、ビデオゲーム、VR 向けに 3D アニメーションや VFX を作成するには、技術的なスキルはもちろん、創造的なスキルも必要です。Houdini は、これらの世界を 1 つにまとめるのに最適なツールです。コンセプトから最終的な仕上げまで、プロジェクトを探求、創造、洗練させていくことができます。

Houdini には、CG コンテンツの生成用に設計されたさまざまなツールが備わっていますが、Houdini ならではの特徴が、**ノードベースのプロシージャルなワークフロー**です。このアプローチなら、**制御可能なショットの作成**、**複数回のイテレーション**、**期限の遵守**の達成が容易になります。Houdini を学習するときには、ノードやネットワークの扱い方を理解することが成功へのカギです。

学習内容

- この「概要」の章では、重要なコンセプトやアイデアを理解するのに役立つ、Houdini の基本情報を紹介します。はじめからすべてを理解するのは難しいでしょう。この章を参照しながら、Foundations チュートリアルを進め、知識を積み上げていってください。
- **3D ソフトウェアの使用経験がある方は**、そこで培ったスキルを活かせます。Scene View とシェルフツールを使ってインタラクティブにショットを構築する方法を学んでから、Houdini のプロシージャルな性質を利用するためのノードやネットワークの使い方に進みます。

3D やコンピュータグラフィックスに初めて触れる方にも、Houdini はぴったりのパッケージです。この学習教材「Foundations」は一定の基礎知識があることを前提としているため、知らない CG の概念については、調べながら読み進めることをお勧めします。Houdini を学習すると、Houdini だけでなく、3D アプリケーション一般について、内部で何がなされているかについての理解が進みます。

「Foundations」のチュートリアルを完了したら、**SideFX.com** には他のチュートリアルもあります。ぜひ挑戦してください メインメニューで **Learn > Learning Paths** を選択すると、SideFX および Houdini コミュニティのメンバーが作成したレッスンの一覧を確認できます。たくさんのレッスンを用意していますので、Houdini のスキルアップにお役立てください。

HOUDINI を無償でダウンロードする

SideFX が提供する学習用体験版で、レッスンを実行できます。体験版 **Houdini Apprentice** なら、Houdini の機能を**無償**で利用できます。ただし、レンダリング解像度やユーザインターフェースの制限、ウォーターマークの追加など、制約事項がいくつかある点に留意してください。

Houdini Apprentice は、SideFX の Web サイトでダウンロードできます。定期的に更新される最新バージョンも、この Web サイトで提供しています。

SideFX.com/download

独立系のアニメータおよびゲーム開発者

学習用体験版では物足りない方には、**Houdini Indie** がお勧めです。Apprentice のようなウォーターマークが追加されないうえ、最大 4K x 4K の高いレンダリング解像度が可能です。商用利用については、**約 1000 万円以下**の収益の用途に制限されます。

この Indie ライセンスにより、Houdini は個人プロジェクトやインディゲームの開発に最適なツールとなっています。詳細は以下をご覧ください。

SideFX.com/indie

ドキュメントバージョン 4.0.1J | 2023 年 8 月 | Houdini 19.5+

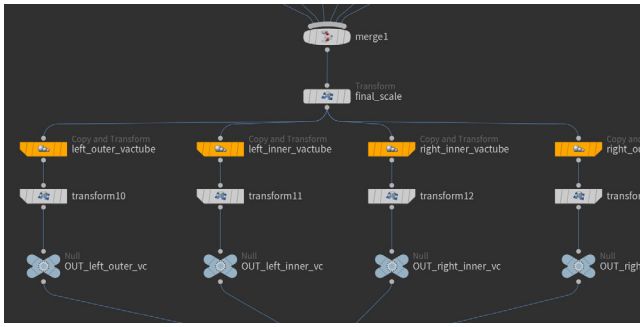
Houdini について

Houdini は、モデリング、アニメーション、レンダリング、シミュレーションに使用できる CG（コンピュータグラフィックス）アプリケーションです。Houdini を学習しながら、ノード、ネットワーク、アセットをインタラクティブに操作するという、制作プロセスの新しい制御方法を探求していきます。

Houdini では、すべてがプロシージャルです。つまり、モデリング、キャラクタリギング、ライティング、レンダリング、ビジュアルエフェクトのすべてにおいて、ノードベースのワークフローの恩恵を受け、ノードネットワークを構築するだけで、クリエイティブなタスクに必要とされるすべての手順を実行できます。ネットワークは他のネットワークと「通信」でき、それによってさらに複雑なシステムを構成します。

プロシージャル

Houdini では、ユーザのすべてのアクションがノードとして格納されます。ノードはネットワークに「接続」され、「レシピ」を作ります。レシピを微調整して繰り返し可能な成果を定義でき、イテレーションごとにユニークな結果を得ることができます。重要な情報をアトリビュートという形で下流に渡すことができるノードの機能が、Houdini のプロシージャルな性質を支えています。



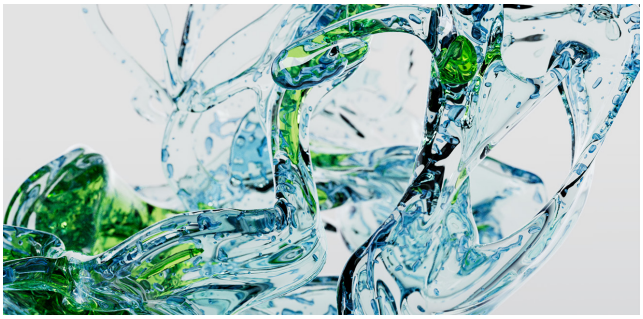
VFX が得意

Houdini がビジュアルエフェクトアーティストを魅了し続けているのは、パーティクルやダイナミクスを扱うには、このプロシージャルなワークフローが理想的だからです。ビジュアルエフェクトはたいてい、ショット内で起きるアクションに反応するよう設計されており、プロシージャルなソリューションがそうした反応を「自動化」します。Houdini を使用すると、スタジオの生産性が向上し、制作プロセスをより詳細に制御できるようになります。

Houdini はまた、大規模なデータセットを扱えるため、リジッドボディの破壊、流体、パーティクルなど、多数のレイヤーが相互作用して最終的な結果が作成される、複雑なビジュアルエフェクトにも対応できます。

プロシージャルな表現

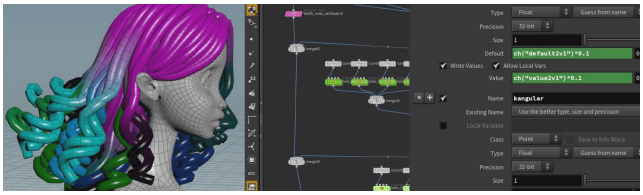
モーショングラフィックスのプロジェクトでは、プロシージャルなアプローチによって、視覚的に魅力的なエフェクトを豊富に作ることができます。こうした特殊効果の多くは、ノード上でパラメータをアニメートし、現実にはありえないような面白い方法でノイズを追加することで得られます。



幅広い CG パイプライン

VFX とモーショングラフィックスだけではありません。Houdini は、モデリングからレンダリング、キャラクタ制作からゲーム開発まで、パイプラインのあらゆる部分に対応できる基本ツールを備えています。Houdini のプロシージャルなワークフローは、あらゆる CG コンテンツの作成をサポートします。その過程では、複数のイテレーションを検討したり、細部にわたって変更を加えることもできます。

ノードは Houdini ならではの特長であり、パワーの根源ですが、ビューポートやシェルフツールも豊富に用意されています。それらをインタラクティブに使用しながら、Houdini でネットワークを構築することができます。

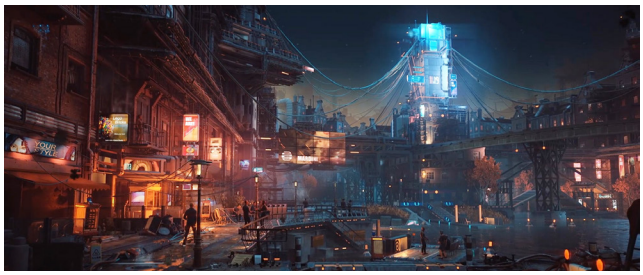


結果の制御しやすさ

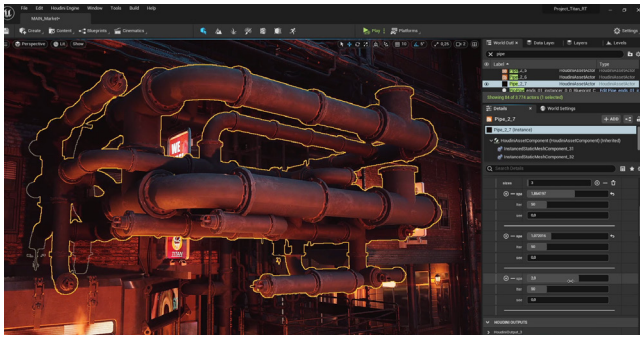
プロダクションの細部まで編集できるのは、Houdini のノードでパラメータに変更を加えると、ネットワークを通じて次々に変更が伝達され、結果が更新されるからです。この制御しやすさは制作プロセス全体で維持されるため、土壇場で決断をしても、従来の CG パイプラインのようなコストはかかりません。

ツールの構築

ノードベースのアプローチのもう 1 つの利点は、ノードネットワークをカプセル化してカスタムノードを簡単に作成できることです。コードを書く必要もなく、同僚と共有できます。Houdini の再利用可能なネットワークは、**Houdini Digital Asset** と呼ばれる特別なノードに、手間をかけずに簡単にラップできます。

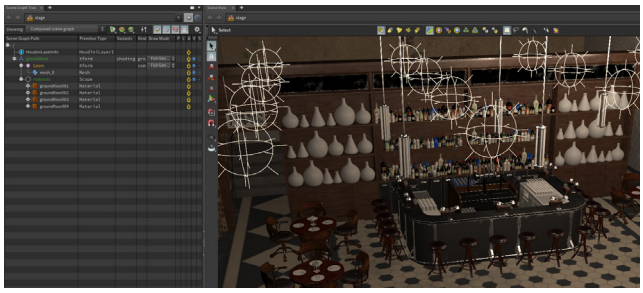


これらのアセットは、Houdini はもちろん、**Autodesk Maya**、**3ds Max**、**Unreal**、**Unity** などの他のアプリケーションでも **Houdini Engine** プラグインを使用して開けます。アセットのプロシージャルな性質もそのまま保たれます。



全データへのフルアクセス

一般的なアニメーションまたはビジュアルエフェクトのパイプラインでは、オブジェクトに情報が蓄積されていきます。通常は Velocity、キャプチャウェイト、UV テクスチャ座標などのポイントまたはプリミティブアトリビュートとして保存されます。他の 3D アプリケーションはこの情報を隠し、舞台裏で制御しようとしますが、Houdini にはこうしたデータを使用および管理できるツールが備わっています。はるかにパワフルで柔軟なアプローチが可能となるため、プロダクション全体が大きく改善します。



新しい考え方

Houdini を使い慣れると、ショットやゲームレベルに新たな方法でアプローチできるようになり、個人およびチームの生産性が向上します。柔軟性に優れた Houdini では、プロジェクトのライフサイクル全体をサポートするツールを構築できます。また、問題や課題に対処するだけでなく、弱点を予測し、さらなる効率化を実現するプロシージャルなソリューションを利用できます。

Houdini を学ぶことは、今後のプロジェクトへのアプローチ方法を再定義するような、多目的なアプリケーションを探求することです。この新しい考え方を受け入れ、想像以上の深いレベルで CG の世界を探求しましょう。

Houdini ではコードを書く必要がありますか？

いいえ、書く必要はありません！ それどころか、他の 3D アプリケーションならコードを書かなければ得られない結果を、Houdini のノードベースのワークフローのおかげでインタラクティブに作れることもよくあります。Houdini はまさにアーティストのためのツールです。スクリプトやエクスプレッションを使用する技術的な側面もありますが、備わっているツールだけで驚くべき成果を得られます。また、ノードの仕組みのおかげで、創作プロセスにつきものの試行錯誤も簡単に行えます。

コードで作業したい方には、Houdini は Houdini インターフェース内で多くの言語をサポートしています。Wrangle ノードでは VEX と Python を使用でき、PyQT もサポートされています。Houdini のエクスプレッション言語である HScript も使用できるうえに、特定のニーズに合わせて混用することも可能です。

Houdini ワークスペース

Houdini のユーザインターフェースは、他の CG アプリケーションを使用してきたアーティストには馴染みやすいでしょう。最大の違いは、ノードとネットワークを管理するペインです。ワークスペースはさまざまに設定可能で、作業方法に合わせてセットアップすることができます。

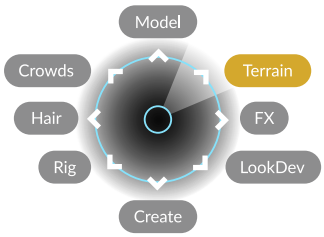
Houdini は、3D シーンを構成する各種要素をさまざまな方法で表示できます。カメラ越しにジオメトリを見る **Scene View** や、プロシージャルノードとネットワークを管理する**ネットワークビュー**などを使用して、各ショットが技術的に機能することを確認しながら、クリエイティブな決定を下すことができます。

Radial メニュー

Houdini のツールにアクセスする方法の 1 つに、**X**、**C**、**V** のホットキーを使ってアクセス可能な Radial メニューがあります。いずれかのホットキーを押すと、Radial メニューが表示され、各種オプションを選択できます。各メニューの主な内容は次の通りです。

- **スナップ** X
- **メイン** C
- **ビュー** V

Radial メニューの仕組みを理解すると、ウィジェットに頼らずに、素早くマウスカーソルを動かすだけでツールにアクセスできます。



デフォルトで **Main** と表示されている、メニューバーの上部の**カスタム**メニューを変更できます。OS X では、これが **Radial** メニューです。

シェルフツール

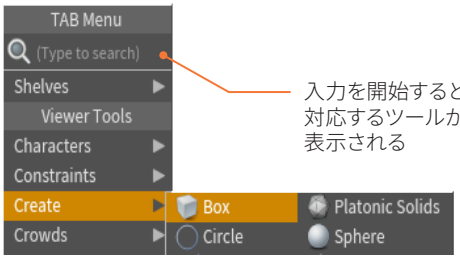
ワークスペースの上部には複数のシェルフがあり、オブジェクト、ジオメトリ、カメラ、ライト、エフェクトを作成したり操作できるツールが多数搭載されています。

これらのツールは Scene View で機能し、多くの場合、何らかの相互作用を伴います。ツールを使用すると、1 つまたは複数のノードが作成され、**パラメータエディタ**や**ネットワークエディタ**で微調整することができます。

シェルフは、Houdini を使い始めたばかりのアーティストには非常に重要なリソースです。クリック回数を減らすことができ、配置されたノードのネットワークからさまざまなことを学べるためです。

タブメニュー

Scene View または**ネットワーク**ビューでツールにアクセスするもう 1 つの方法が、**Tab** キーです。利用可能なツールやノードのメニューが表示されます。



ツールシェルフ - シェルフツールを使用すると、Scene View でオブジェクトやジオメトリを操作できます。

ツールバー：

選択モード - シーン、ジオメトリ、またはダイナミックオブジェクトにフォーカスします。

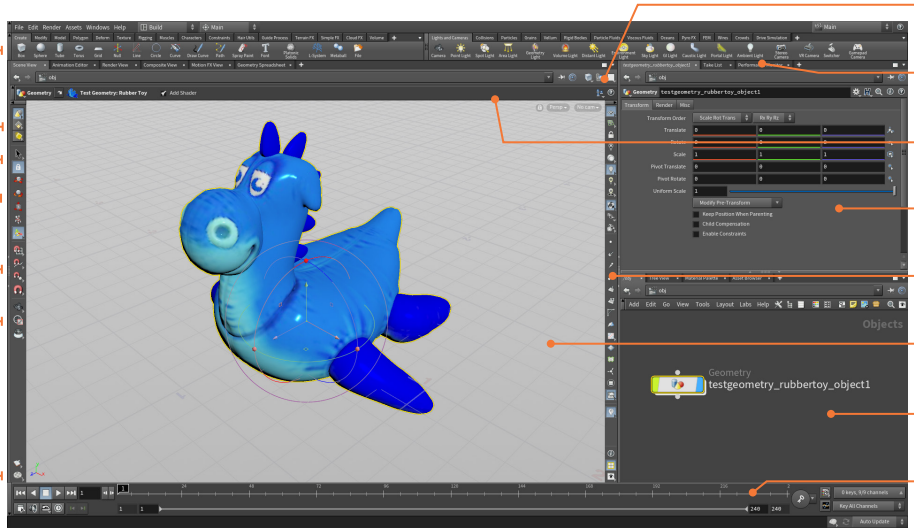
選択ツール - Select、Secure Selection。

トランスフォームツール - ノード固有のコントロール用の Move、Rotate、Scale、Pose、または Handle ツール。

スナップツール - グリッド、プリミティブ、ポイントへのスナップと複数スナップをオンにします。

ビューイング - View ツールを使用してタンブル、パン、ドリーしたり、Render Region ツールを使用して Scene View でレンダリングできます。クリックしたままにすると、2D パン/ズームに変更できます。

出力ツール - これらのツールを使ってシーンをレンダリングしたり、フリップブックを作成できます。



3D View ツール

ビュー操作で使用できるホットキーの組み合わせをいくつか紹介します。実際に **View** ツールを使用している間は、**スペースバー/ Alt** を省略できます。

- **タンブル** スペースバーまたは Alt (Opt) + 左マウスボタン (LMB)
- **パン** スペースバーまたは Alt (Opt) + 中マウスボタン (MMB)
- **ドリー** スペースバーまたは Alt (Opt) + 右マウスボタン (RMB)

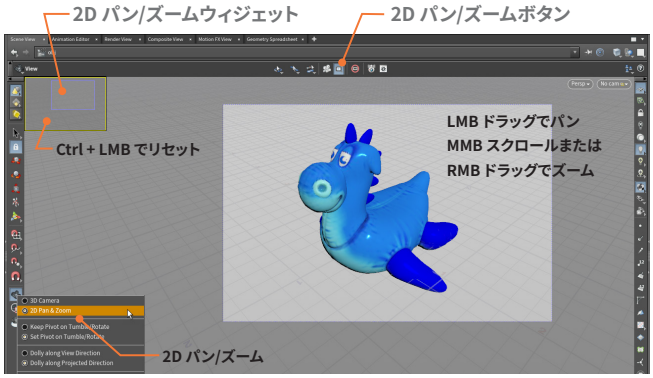
View ツールはツールバーにあります。**スペースバー**または **Alt** キーを押すと、一時的に View ツールを呼び出すことができます。ビューで選択または操作しているときに、視点を素早く変更したい場合には非常に便利です。

ビューイングに集中したい場合は、**Escape** を押すと **View** ツールに移動できます。ビューの操作に使用できる便利なホットキーをいくつかご紹介しておきましょう。

- **Home Grid** スペースバー + H
- **Home All** スペースバー + A
- **Home Selected** スペースバー + G

2D パン/ズーム

オペレーションコントロールツールバーにある **2D パン/ズーム**ツールをクリックすると、3D カメラの位置を変えることなく、2D でのビューを変更することができます。左上のウィジェットでは、クリックしてパンやズームしたり、**Ctrl + LMB** クリックでビューをリセットできます。これは、ロックされたカメラで作業する際に便利なツールです。



ビューポートディスプレイメニュー - オブジェクトの表示方法やビューの構成を変更できます。

ペインタブ - これらのタブを使用すると、複数のパネルを同時に作成したり整理できます。

オペレーションコントロール - このツールバーと Handle ツールを使用すると、選択したノードのパラメータにアクセスできます。

パラメータエディタ - 選択したノードに値を設定したり、エクスプレッションを追加したり、キーフレームを設定することができます。

Display Options バー - 法線、ポイント番号、ライティングといったシーンのディスプレイオプションを制御できます。

Scene View - 作業内容を視覚化したり、ハンドルを使ってシーン内のオブジェクトをインタラクティブに操作できます。

ネットワークエディタ - ノードのネットワークを表示したり管理して、シーンの基礎構造を操作できます。

プレイバー - 選択したノードで現行時間を設定したり、キーフレームを編集できます。プレイバーを使用すると、キーフレームをコピーアンドペーストすることもできます。

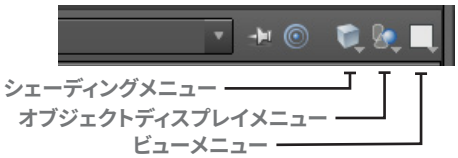
一人称視点のカメラ

View ツールでは、ビデオゲームで目にするような、一人称視点のフライスルーモードをオンにできます。

- **一人称視点のオン/オフ** M
- **ドリーイン/ドリーアウト** W/S
- **右にパン/左にパン** A/D
- **視点を回転** LMB

ビューポートディスプレイメニュー

Scene View の右上にあるメニューまたは **V** Radial メニューを使用して、オブジェクトの表示方法やビューの構成を変更できます。



シェーディングメニュー - Wireframe、Flat Shaded、Smooth Shaded、Smooth Wire Shaded などのオプションがあります。

オブジェクトディスプレイメニュー - ネットワーク内に入ったとき、ジオメトリを非表示にするか、表示するか、ゴースト表示するかを設定します。

ビューメニュー - Scene View をパースビューや正射投影ビューなどのさまざまなビューに分割できます。

Display Options バー

Scene View の右側にある Display Options バーでは、ビューポート表示のオプションにアクセスできます。いくつかをご紹介しましょう。

Reference Plane/Ortho Grid - 参照用やグリッドスナッピングに使用できるグリッドをオン/オフします。

Construction Plane - オブジェクトやポイントの配置場所を定義するのに使用されるコンストラクション平面をオン/オフします。

Lock Camera - 現在のカメラをビューに固定して、ビューを変更するとカメラのトランスフォーム値も変更されるようにします。

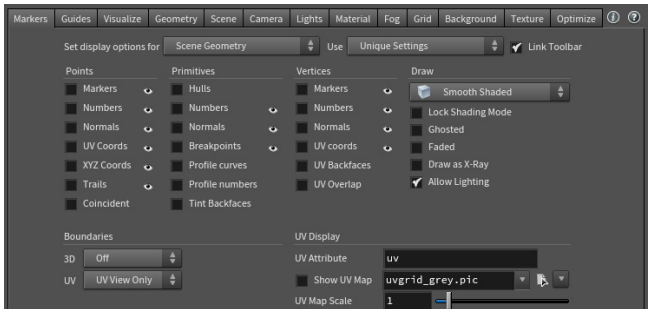
High Quality Lighting with Shadows - ビューポートレンダリングの最高画質を設定します。

Display Primitive Normals - シーンのすべてのプリミティブに属する法線を表示し、その方向を特定します。

Display Options

Scene View とネットワークビューには、それぞれ Display Options パネルがあり、Display Options バーの下部にあるアイコンをクリックすることでアクセスできます。

Display Options D

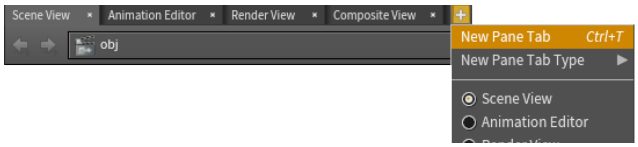


ペインとデスクトップ

Houdini ワークスペースは、シーンデータをユニークな方法で編成した、いくつかのペインに分かれています。3D ビューでインタラクティブに作業したり、スプレッドシートでアトリビュート値を分析することもできます。こうしたさまざまな UI 要素を有効に利用する方法を学ぶことが重要です。

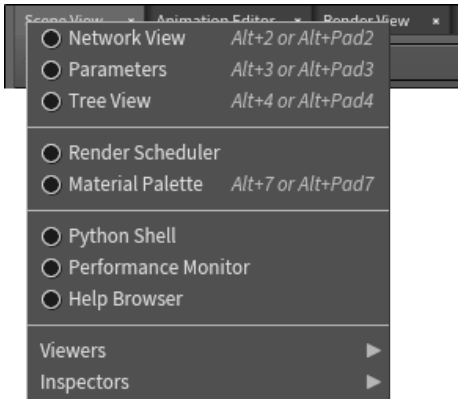
ペインとペインタブ

Houdini ワークスペースはペインに分割されており、シーンをセットアップしたり探求できるようになっています。ペインタブを使用すると、同じゾーン内に複数のペインをオーバーラップさせて、便利な状態に保つことができますが、デフォルトでは表示されません。
ワークスペースでペインタブをクリックすると、そのペインタブにアクセスできます。閉じるには X をクリックします。**+（プラス）メニュー**を使用すると、表示されているペインを変更したり、新しいペインを追加することができます。



ペインタイプ

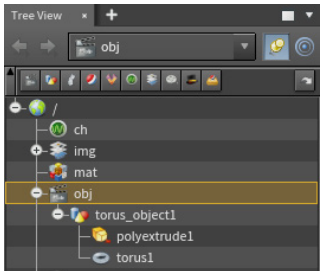
ペインタブを **RMB クリック**して、タイプを変更します。たくさんのペインタイプから選択できます。一部のタイプにはホットキーがあります。他のタイプのいくつかも、ここにリストしました。すべてのタイプについて詳しく確認するには、別途ドキュメントを参照してください。



Network View (Alt + 2) - このビューでは、ノードやネットワークを確認したり、ニーズに合わせて接続、再接続、再編成を行うことができます。

Parameters (Alt + 3) - パラメータに値を設定したり、エクスプレッションを追加したり、ノードのプロパティを制御することができます。

Tree View (Alt + 4) - ノードを階層表示します。これは、シーン階層を理解するのに便利です。



Viewers > Scene View (Alt + 1) - 3D 空間でインタラクティブに作業できます。このタイプのビューでは、1 つまたは複数のビューポートをセットアップできます。複数の Scene View パネルを同時に開いて、さまざまな視点からシーンを見ることが可能です。

Composite View (Alt + 0) - Compositing（COP）ノードを使用して作成された画像やコンポジットを表示します。

Viewers > Motion FX View (Alt + ^) - Houdini のチャンネルオペレータ（CHOP）ノードを使用して作成されたモーションを表示します。

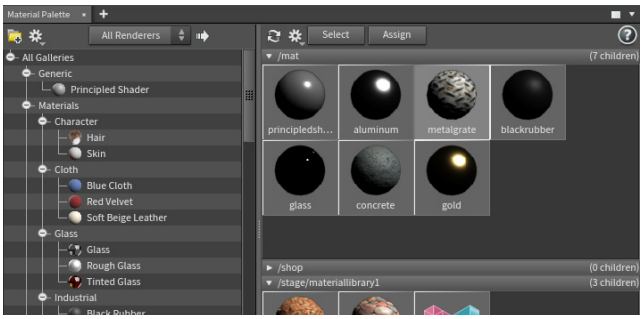
Solaris > Scene Graph - Solaris（LOP）ノードを使用する際、USD シーングラフを表示します。

Solaris > Render Gallery - テストレンダリングを保存し、すべての画像を確認した後、各画像の設定に戻すことができます。

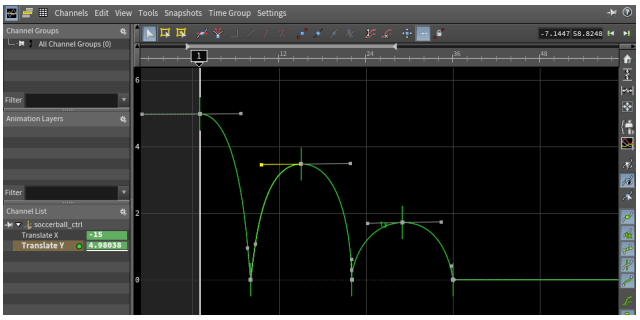
Solaris > Light Linker - ライトとオブジェクトを接続します。

Render Scheduler - 完了したレンダリングや進行中のレンダリングを表示します。レンダリングを停止したり、強制終了することができます。

Material Palette (Alt + 7) - シーン内のすべてのマテリアルを表示し、オブジェクトやジオメトリに割り当てることができます。



Animation > Animation Editor (Alt + 6) - キーフレームとアニメーションカーブを管理できます。アニメーションエディタには、**テーブルビュー**と**ドープシートビュー**もあります。



Animation > Channel List - チャンネルグループを作成し、Houdini でアニメートする際にスコープされたチャンネルを管理できます。

Animation > Autorigs - 二足リグ、四足リグ、フェイシャルリグ用に、独自のリグをモジュールから構築できるツールにアクセスできます。

Animation > Character Picker - このペインを使用すると、キャラクタリグのパーツを簡単に選択できるようになります。

Inspectors > Geometry Spreadsheet (Alt + 8) - ジオメトリのアトリビュート値を表示します。UV、法線、独自に設定したカスタム値などが含まれます。

Inspectors > Data Tree - Light Bank、Material Stylesheet、Object Appearance エディタにアクセスできます。

Mantra Rendering > Render View (Alt + 9) - インタラクティブな Mantra レンダリングを開始します。シーン内で変更を加えると、更新されます。

Mantra Rendering > Take List - 特定のパラメータを変更することで、さまざまな「テイク」を探求できます。テイクを管理して、気に入ったものにフォーカスできます。

TOPS > Task Graph Table - グラフ内のすべてのワークアイテム、または特定のノードのすべてのワークアイテムのメタデータを表示します。

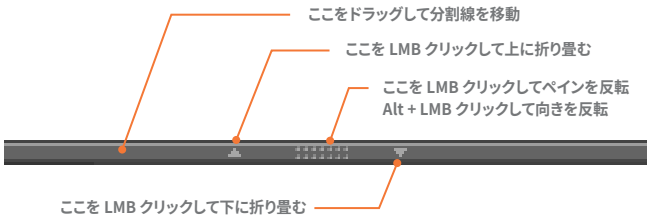
Misc > Orbolt Asset Browser - Orbolt.com のアセットにアクセスできます。このペインを使用するには、orbolt.com アカウントでログインする必要があります。

Misc > Textport - コマンドを入力できます。

Misc > Python Shell - Python コマンドを入力できます。

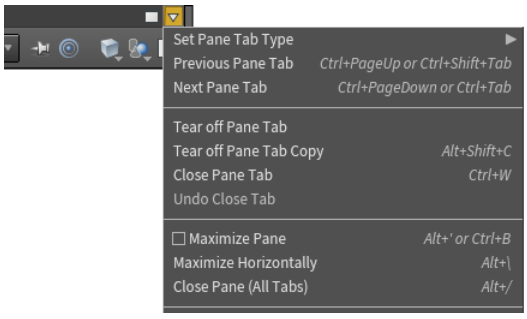
ペインの整理と折り畳み

ペインとツールバーは、それぞれの UI にある矢印をクリックすることで、折り畳んだり展開することができます。ペイン全体は左右に折り畳むことができ、中央のグリップを使用すると、内容を反転させることもできます。これらのオプションにより、マウスを 1 回クリックするだけで他のペインを非表示にし、特定のペインにフォーカスすることが可能です。



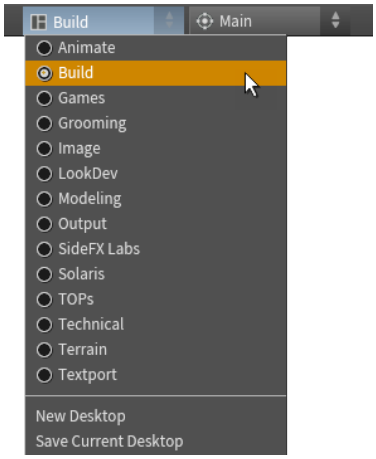
ペインメニュー

各ペインの左上には、ペインを最大化、最小化するためのボタンと、ペインメニューにアクセスするための矢印が表示されています。このメニューでは、ペインやペインのコピーを切り離したり、ペインの削除や分割を行うことができます。また、各ペインの UI を決めるためのオプションもあります。



デスクトップ

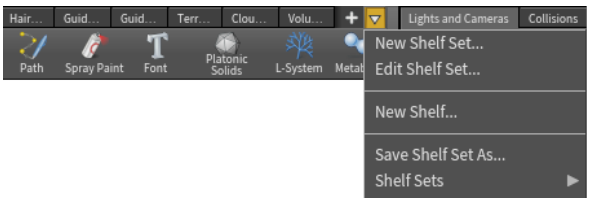
タブを開き、分割線を追加し、ペインタブを整理していくと、自分なりのワークスペースをセットアップできます。レイアウトを保存するには、**Desktop** メニュー (OSX では **Windows > Desktop**) を使用します。このメニューでは、保存したデスクトップにアクセスしたり、独自のデスクトップを保存したり、作業中にデスクトップを管理することができます。デスクトップを保存すると、ペインレイアウト、Radial メニュー、表示されているシェルフセットが保存されます。



シーンを保存した場合、どの**デスクトップ**を見ているかは記憶されますが、作業中にペインレイアウトに加えられた変更は記憶されません。明示的にデスクトップに保存するか、新しいデスクトップを作成しない限り、これらの変更は消えてしまいます。

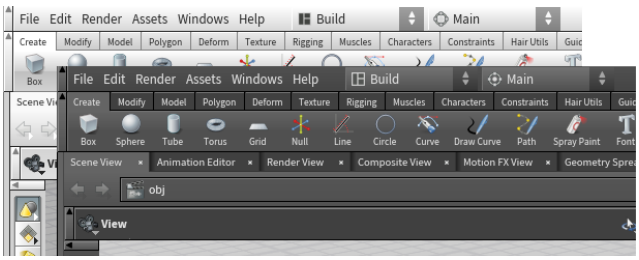
シェルフとシェルフセット

ワークスペースの上部にあるシェルフを管理するには、矢印アイコンの下に表示されるメニューにアクセスします。このメニューを使用して、シェルフセットを操作できます。また、非表示になっているシェルフセットをデスクトップに表示させることもできます。



カラー設定

ワークスペースのカラースキームを選択することで、Houdini UI の外観をカスタマイズできます。**Edit > Color Settings** を選択してオプションウィンドウを表示し、デフォルトの **Houdini Dark** または **Houdini Light** を選択します。**ダウンロード**ボタンをクリックして、Houdini コミュニティで作成されたカラースキームのリストから選択してもかまいません。



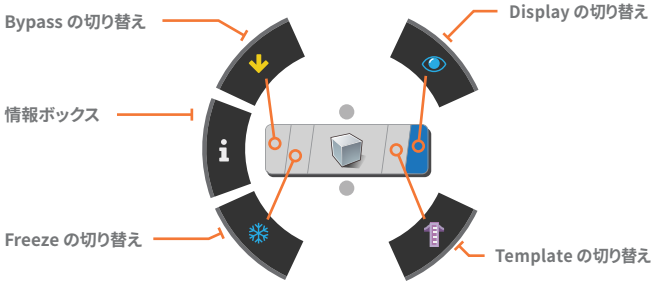
ノードとネットワーク

Houdini のノードベースのワークフローは、プロシージャルなアーキテクチャの中核です。Houdini を有効活用するためには、これらのノードやネットワークを直接操作できることが非常に重要です。ノードというと専門的に聞こえるかもしれませんが、実際には非常にアーティストフレンドリーで、使いやすい仕組みです。

Houdini でツールを使用すると、ノードが作成され、他のノードに接続されます。できあがったネットワークでは、ユーザアクションの履歴が分かるうえ、変更や微調整を簡単に加えられます。Houdini で作業するうえでは、ノードネットワークの効果的な使用方法を学ぶことが重要です。

ノードフラグ

各ノードには、表示されているか、ロックされているか、バイパスされているかを表すさまざまなフラグがあります。フラグそのものをクリックするか、Radial ノードメニューを使用して、フラグを設定できます。



Display フラグ (R) - このフラグを使用すると、表示させる出力ノードを選択でき、ノードは中空のリングでハイライトされます。

Render フラグ (T) は、レンダリングに出力されるノードを設定します。そのノードは塗りつぶされた円でハイライトされます。Display フラグとは別にこれを設定するには、Display フラグを **Ctrl クリック**します。

Template フラグ (E) - このフラグは、ノードをグレーで表示します。参照やスナップに使用することができます。

Freeze フラグ - ロックされたノードでキャッシュ化します。ネットワークがクックされるとき、チェーン内の前のノードはすべて無視されます。

Bypass フラグ (B) - ネットワークがクックされるとき、このフラグを設定したノードは無視されます。

ノードの接続と接続解除

ビューポートで作業するとき、多くの場合、ノードは自動的に配置、接続されます。ネットワークを再編成したい場合は、ノードの接続と接続解除を手動で行う必要があります。

ネットワークエディタでノードを扱ったり接続する方法をいくつか紹介します。

- **ノードの接続** 出力から入力を LMB ドラッグ
- **複数ノードの接続** J を押しながら複数のノード間をドラッグ
- **新しいノードの挿入** 出力またはワイヤー上を RMB
- **ノードの挿入** LMB ドラッグしてワイヤー上にドロップ
- **ワイヤーからの接続解除** LMB でノードを選択して振る
- **ワイヤーの切断** Y を押しながらワイヤーを横切るようにドラッグ
- **ノードの移動** LMB ドラッグ
- **選択したノードのコピー** Alt + LMB ドラッグ
- **参照コピー** Alt + Shift + Ctrl + LMB ドラッグ

ドットを使ってネットワークを整理できます。

- **ドットの追加** ワイヤーを Alt + LMB
- **ドットのピン留め / ピン解除** ドットを Alt + LMB

ノードギャラリー

ギャラリーでは、ネットワークに直接追加したいノードに素早くアクセスすることができます。ギャラリーには、日常の作業でよく使用するノードが含まれており、すべてのノードに **Tab** キーでアクセスできます。**Windows > Gallery Manager** を使用して独自のギャラリーを作成したり、ノードを **RMB クリック**してから **Save to Gallery...** を選択することで、ギャラリーにアイテムを追加できます。

Mat ネットワークに保存されたノードは、Mantra マテリアル用の ***Mantra*** のように適切なキーワードが用いられている限り、**Material Palette** でも利用可能です。

ネットワークビュー

ネットワークパス - 現在のネットワークレベルまでのパス。このバーを使用して他のネットワークに移動することも可能です。

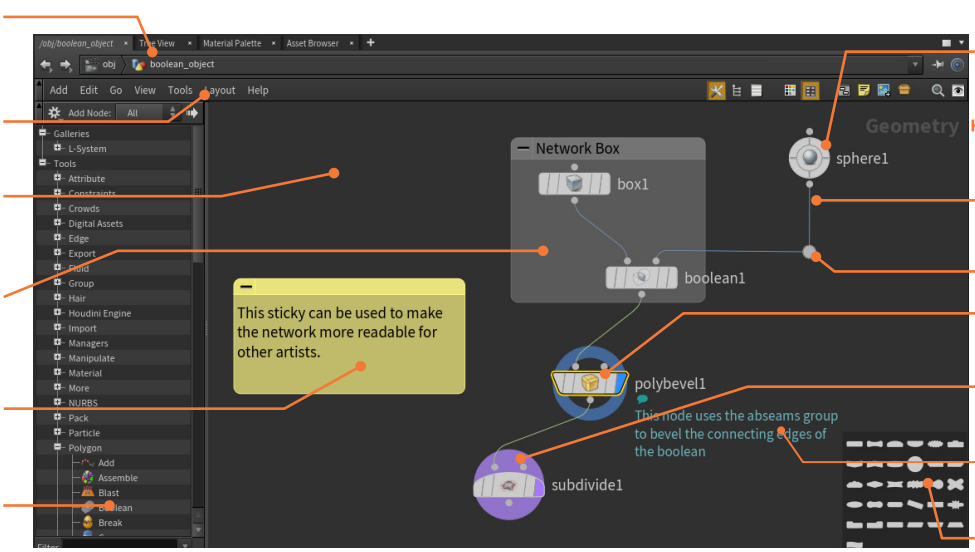
ペインメニュー - ネットワークを整理するためのメニューとアイコン。

ネットワークの背景 - ペインメニューを使用して、画像を追加したりグリッドをセットアップし、ノードを整理しやすくすることができます。

ネットワークボックス - 関連するノードをグループ化します。素早く折り畳んだり展開することができます。

ステッキーノート - メモを追加して、他のアーティストがネットワークを簡単に理解できるようにしたり、説明を加えたりします。

ノードギャラリー - ここからネットワークにノードをドラッグできます。下部のフィルタを使用して、必要なノードを見つけられます。



ネットワークタイプ

Houdini にはさまざまな種類のノードがあり、それぞれが独自のコンテキストで機能します。ネットワークタイプは、ネットワークビューの右上に表示されます。各タイプのノードは、他のネットワークに接続することができます。ノードタイプが違っても接続方法は同様ですが、それぞれ固有の働きをします。

シーン	**オブジェクト**	OBJ
ジオメトリ	**サーフェスオペレータ**	SOP
Solaris	**ライティング / レイアウトオペレータ**	LOP
マテリアル	**VEX Builder**	MAT
Motion FX	**チャンネルオペレータ**	CHOP
VEX	**VEX Builder**	VOP
出力	**レンダリングオペレータ**	ROP
タスク	**タスクオペレータ**	TOP
ダイナミクス	**ダイナミクスオペレータ**	DOP
コンポジット	**コンポジットオペレータ**	COP/IMG

Houdini を使い続けるうちに、このノードタイプを語る「秘密」の言語を理解し、それをプロシージャルに適用する方法が分かるようになるはずです。

ネットワークパス

ノードは階層的に編成されており、ネットワークマネージャまたはサブネットワークと呼ばれる、別のノードにネスト化されている場合もあります。これらの階層を管理しやすくするために、ほとんどのペインの上部にはブラウザのようなパスが表示されています。

このパスを使用して、階層を上下に移動したり、他のネットワークに移動することができます。デフォルトでは、Scene View で選択するとパスが変化しますが、パスを**ピン留め**してフォーカスを保持することも可能です。また、ピン留めしたペインに**ターゲット**アイコンをドラッグし、パスを同期させることもできます。

ネットワークナビゲーション

ネットワークタイプ間の移動には、さまざまな方法があります。Scene View でオブジェクトを操作しながら自然の流れで行う方法もあれば、素早く移動できるショートカットカットもあります。

選択モード - Scene View で選択すると、ネットワークエディタが選択した位置にジャンプします。選択モードによって、選択する際のネットワークタイプが変化します。

ネットワークパス - 親ノードを **LMB クリック**して、上のレベルに移動できます。コンテナノードを **LMB クリック**すると、同じレベルのノードにアクセスしたり、他のコンテナノードの中に入ることができます。

Radial メニュー - N を押して Radial メニューを開き、ネットワーク内を上下に移動したり、別のネットワークタイプに移動することができます。

ホットキー - 以下のホットキーを使ってネットワーク内を移動しながら、選択したオブジェクトを使用できます。

ノードの中に入る	I
上のレベルに移動する	U
オブジェクト / ジオメトリを切り替える	F8
前または次のネットワークに切り替える	Alt + ← または Alt + →

クィックマーク - ネットワークロケーションを設定して、素早くそのロケーションに戻ることができます。必要に応じて使用でき、後で上書きも可能です。忘れても問題ありません。シーンファイルには保存されません。

クィックマークの設定	Ctrl + 1, 2, 3, 4 または 5
クイックマークに戻る	1, 2, 3, 4 または 5
前のビューに戻る	`

選択と表示のホットキー

ネットワークエディタでは、パンやズームしながらネットワーク全体を操作する必要があります。これらのアクションに使用するキーの組み合わせを紹介します。

パン	MMB
ズーム	RMB
ノードの選択	LMB
追加選択	Shift + LMB
選択解除	Ctrl + LMB

ノード - ネットワークの最終出力に寄与するオペレーションを表します。

ネットワークタイプ - どのネットワークタイプで作業しているかを示します。

ワイヤー - どのようにノード同士がつながり、どのようにデータがネットワーク内で渡されているかを示すラインです。

ドット - ドットを追加すると、ノードを整理しやすくなります。

表示リング - この小さい円は、Scene View にどのノードが表示されているかを示します。

レンダーリング - この大きい円は、表示されているノードに関係なく、レンダリングノードを示します。

コメント - 他のアーティストがネットワークを理解しやすいよう、ノードのコメントを表示できます。

パレット - メニューのボタンを使用して、ノードの色や形状を設定できるパレットを表示できます。

ノードを詳しく知る

Radial メニューを使用するか、ノードを **MMB クリック**して**情報ボックス**を表示します。このパネルでは、ノードの内容、グループ、アトリビュートに関する情報や、その他の重要事項について確認できます。また、ワークフローに影響しているエラーも表示されます。

このパネルは自動的に閉じますが、**ピン**アイコンをクリックすると、作業中も表示したままにできます。このパネルを使ってコメントを追加し、ネットワークビューに表示させることができます。

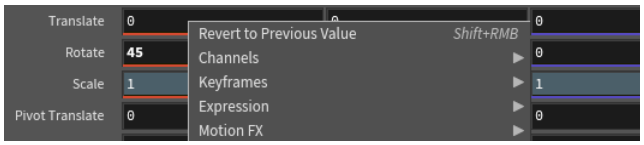
パラメータ、チャンネル、アトリビュート

Houdini のすべてのノードは、希望の結果が得られるよう、パラメータ、チャンネル、アトリビュートによって駆動されます。Houdini で使用されている用語は、他の 3D アプリケーションとは異なる場合があります。時間をとって、Houdini での用語の意味を理解することをお勧めします。

パラメータ

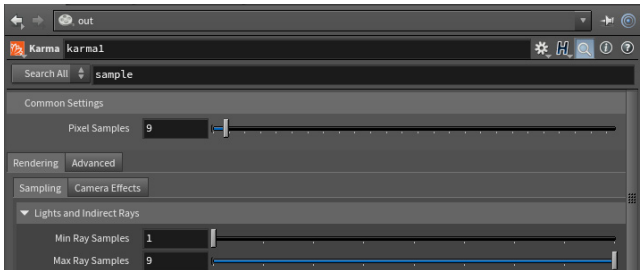
パラメータとは、Houdini ノードの値、スライダ、ボタン、チェックボックスのことを指します。これらは他のアプリケーションでアトリビュートと呼ばれる場合もありますが、Houdini ではアトリビュートを別の意味で使用します。

パラメータ値は、パラメータエディタで、またはビューポートでハンドルを使用して変更することができます。各パラメータには RMB メニューがあり、コピー、ペースト、デフォルトに戻すといった重要なオプションが多数用意されています。



パラメータの検索

ノードには多数のパラメータがあり、すべてを見ていくのは時間がかかります。右上の虫眼鏡をクリックすると、検索バーが表示され、名前と内容に基づいてパラメータをフィルタリングできます。エクスプレッション、オーバーライド、さらにはパラメータ値そのものを使用して、パラメータを見つけることができます。



チャンネル | キーフレーム

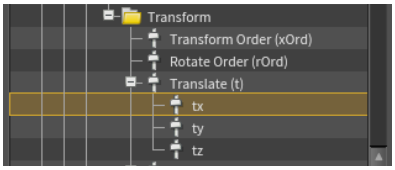
パラメータにキーフレームを設定するには、**Alt** キーを押しながら、名前または値のフィールドを **LMB クリック**します。キーフレームを設定すると、パラメータのフィールドの色が変わり、アニメートされたチャンネルが作成されます。キーフレームがパラメータに関連付けられ、アニメーションエディタでアクセスできるようになります。

チャンネル | エクスプレッション

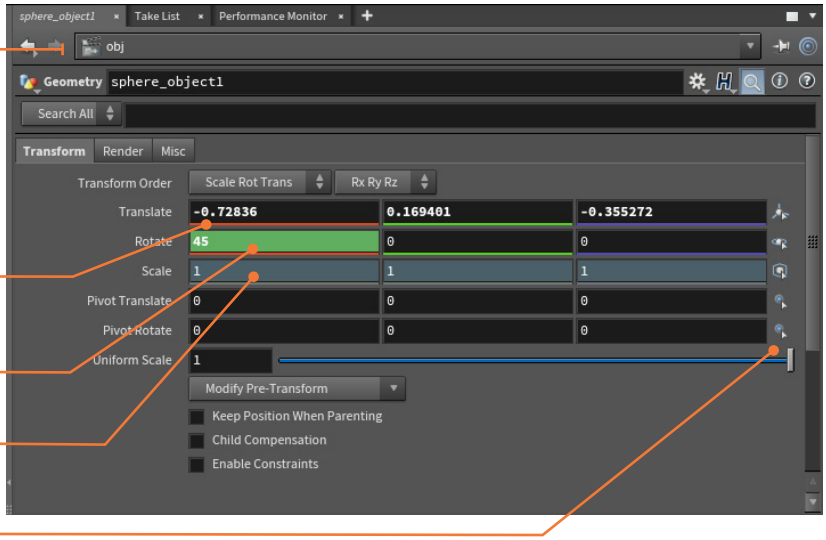
値そのものではなく、***hScript*** または ***Python*** を使用して、パラメータにエクスプレッションを追加することもできます。パラメータエディタの右上に、使用する言語を選択できるメニューがあります。**Ctrl + E** を押すと、スクリプトツールを多数備えたエクスプレッションエディタが表示され、作業が容易になります。

シーンデータの参照

パラメータを **RMB クリック**して、**Reference > Scene Data** を選択すると、参照したいものを具体的に選択できるウィンドウが表示されます。シーン内の任意のノードから選択して、チャンネル参照を作成できます。この方法なら、正確な構文で適切なエクスプレッションを書く苦労なしに、参照を作成できます。



パラメータエディタ



ナビゲーションバー - このバーでは、ノードがシーン階層のどこに位置しているかを確認できます。

ノードのタイプと名前 - ノードタイプとノード名が表示されます。アイコンをクリックすると、ノードを使用するためのメニューが表示されます。

検索バー - 虫眼鏡のアイコンをクリックすると、名前や内容でパラメータを検索できます。

パラメータの変更 - パラメータがデフォルトから変更されると、値が太字で表示されます。フォルダタブ名も太字になります。

アニメーションパラメータ - パラメータにキーフレームを設定すると、緑でハイライトされます。

パラメータのロック - パラメータを RMB クリックして、ロックしたりロック解除できます。グレーでハイライトされます。

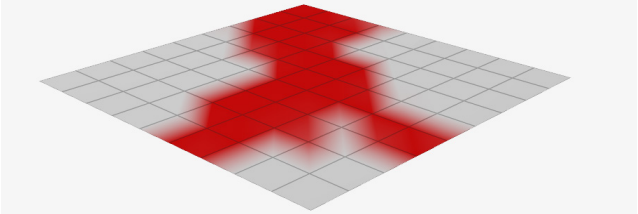
選択してマッチ - これらのアイコンを使用すると、パラメータ値を他のオブジェクトにマッチさせることができます。

カスタムパラメータ

パラメータエディタの右上にあるギアアイコンをクリックすると、**Edit Parameter Interface** を選択できます。このウィンドウではカスタムパラメータを追加して、ノードネットワークの他の部分にリンクすることができます。

アトリビュート

アトリビュートを使うと、ジオメトリにデータを取り付けられます。そのデータは、オペレーションを完了するまでチェーン内のノードで順に使用されます。fuel アトリビュートは Pyro FX シミュレーションを駆動し、UV アトリビュートはテクスチャリングをセットアップします。Houdini ノードによって作成されるアトリビュートも、独自に作成するカスタムアトリビュートもあります。

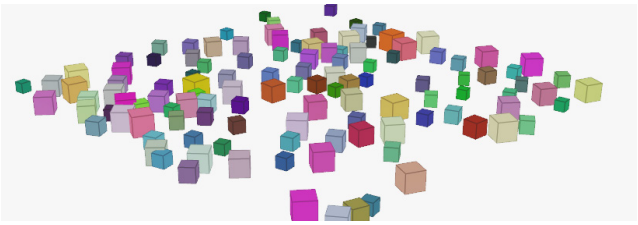


クラス - アトリビュートは、ポイント、プリミティブ、ディテール、頂点に属します。これは、チェーンでどのように使用されるかに影響します。

タイプ - 浮動小数点、整数、文字列のアトリビュートタイプをセットアップできます。

ATTRIBUTE RANDOMIZE

Attribute Randomize を使用すると、アトリビュートを作成し、その値を即座にランダム化することができます。例えばこの例では、ボックスの色、回転、スケールがランダム化されています。

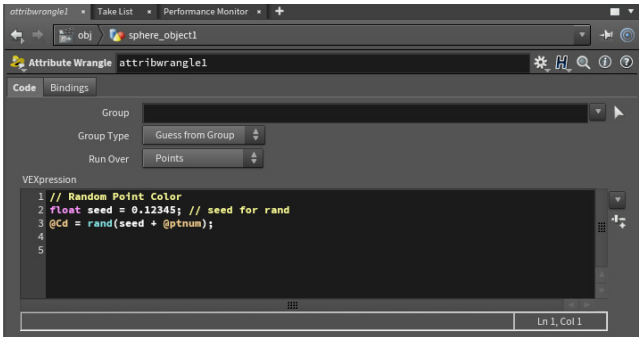


Attribute Transfer

ノードチェーン内で、アトリビュートはジオメトリに取り付けられ、他のノードでも使用されます。**Attribute Transfer** を使用すると、他のジオメトリピースにアトリビュートを渡すこともできます。この例では、球が定義された閾値に基づいて、ボックスにカラーアトリビュートを渡しています。

ATTRIBUTE WRANGLE

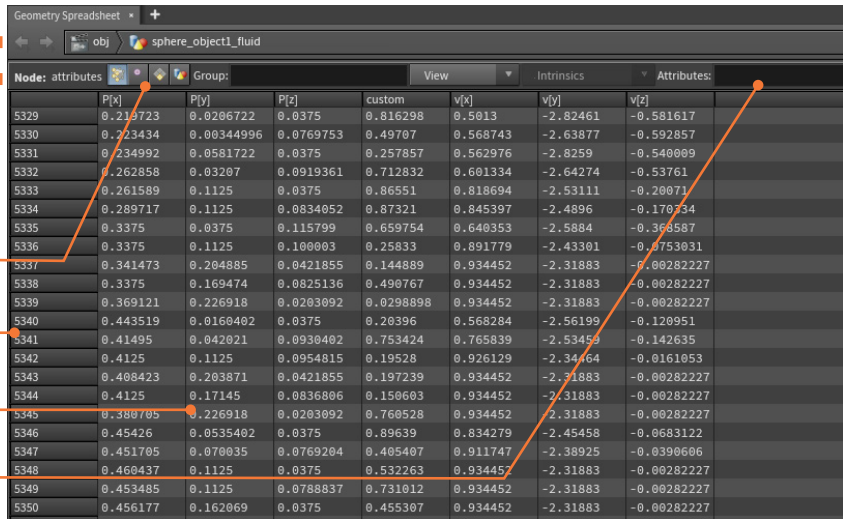
Houdini には、アトリビュートを作成したり使用するための多様なノードがあります。**Attribute Wrangle** ノードを使用すれば、スクリプトベースのアプローチを取ることも可能です。これは、多くのテクニカルディレクタが非常にやりやすいと感じる方法でしょう。



アーティストにとっては、ノードを使用した方がこうした情報を扱いやすいものです。Houdini を有効活用するには、アトリビュートを適切に使用することが重要であり、いずれはアトリビュートについて学ぶ必要がある時がきます。

Geometry Spreadsheet

Geometry Spreadsheet では、非表示のものも含め、すべてのアトリビュート値を確認できます。



ナビゲーションバー - このバーでは、ノードがシーン階層のどこに位置しているかを確認できます。

ノード名 - どのノードが現在選択され、これらのアトリビュート値を生成しているかを示します。

Attribute Class ボタン - これらのボタンを使用すると、表示する属性の種類をフィルタリングできます。

ポイント番号 - ジオメトリのポイント番号は、モデル上でのアトリビュートの位置を特定しやすくします。

アトリビュート値 - ノードネットワークチェーンのこのポイントでの値です。

フィルタ - 多くのパラメータを使用している場合は、ここにパラメータ名を入力すると、リストをフィルタリングできます。

ジオメトリの選択

Houdini での作業では、さまざまな要素を選択して操作する必要があります。ポイント、エッジ、プリミティブなどのオブジェクトやジオメトリコンポーネントを効率的に扱うためのツールやオプションが多数用意されています。

Select ツール

Select ツールを使用すると、選択に集中でき、操作ハンドルは表示されません。

- **Select ツール** S

Move や **Rotate** ツールを使用する場合や、**Secure Selection** がオンのときは、**Select** ツールを呼び出して選択する必要があります。**Secure Selection** を**オフ**に切り替えると、自由に選択できるようになります。

- **他のツールの使用中に Select ツールを呼び出す** S を押したままにする
- **Secure Selection の切り替え** ~

選択タイプ

追加選択、選択解除、選択/選択解除の切り替え、すべて選択、なにも選択しないなど、さまざまホットキーがあります。これらのテクニックは、Houdini ワークフローで重要な役割を果たします。

- **選択** LMB
- **追加選択** Shift + LMB
- **選択解除** Ctrl (Cmd) + LMB
- **選択/選択解除の切り替え** Ctrl (Cmd) + Shift + LMB
- **すべて選択** A (オブジェクトレベル) / N (ジオメトリレベル)
- **なにも選択しない** N (オブジェクトレベル) / Shift + N (ジオメトリレベル)

選択テクニック

ビューポートでは、4 種類の選択タイプのいずれかでジオメトリにアクセスできます。

- **Box Selection** F2
- **Lasso Selection** F3
- **Brush Selection** F4
- **Laser Selection** F5

選択フィルタを使用して、可視のジオメトリにフォーカスしたり、グループを選択したりすることもできます。選択を容易にするオプションが、多数用意されています。

- **Select Visible Geometry Only** Shift + V
- **Select Fully Contained Geometry Only** Shift + C
- **Select Groups または Connected Geometry** 9
- **Select Whole Geometry** オペレーションコントロールツールバーで選択
- **Select by Normals** オペレーションコントロールツールバーで選択

選択モード

選択モードを使用すると、オブジェクトやコンポーネントを選択できるようになります。ツールバーのボタンやホットキーを使用すれば、オブジェクトレベルからジオメトリレベルに簡単にジャンプすることも可能です。

オブジェクト - オブジェクトネットワークレベルは、オブジェクトのトランスフォームを操作する場所です。**View** ツール以外のツールでは、次のホットキーでオブジェクトレベルに戻ることができます。

- **オブジェクト** 1

ジオメトリ - **View** ツール以外では、次のいずれかのホットキーを使用すると、選択したコンポーネントが選択可能な状態になっているジオメトリレベルにジャンプすることができます。

- **ポイント** 2
- **エッジ** 3
- **プリミティブ (フェース)** 4
- **頂点** 5

TWEAK モード

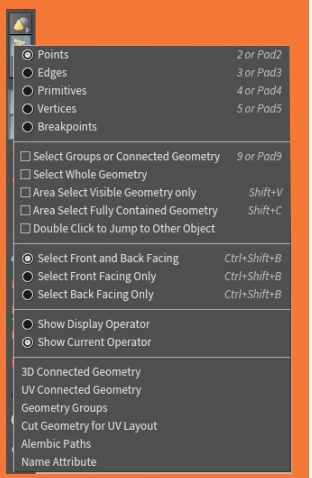
同時にアクティブにできるジオメトリ選択モードは、1 つだけです。**Edit** ノードで作業している場合は、**Tweak モード**でポイント、エッジ、プリミティブの組み合わせを選択することができます。

選択モードのメニュー

各選択モードには、シーンにどのように作用するかを変更できるオプションが用意されています。これらのオプションにアクセスするには、各モードのアイコンを **LMB クリック**または **RMB クリック**します。

コンポーネントを扱う場合、このメニューでは **Show Display Operator** または **Show Current Operator** を選択できます。**Edit** ノードを使用しているとき、これらのオプションは Scene View の上部でも利用可能です。

オブジェクトレベルでは、このメニューは異なります。オブジェクトの種類に応じたフィルタや、マテリアル、拘束、デジタルアセットをより簡単に選択するためのオプションが含まれています。



選択オプション

編集 | コンポーネント - これらのボタンから、どのコンポーネントを作業するかを選択します。ここではエッジ選択がオンになっています。

Select ツール - **Select** ツールを使用して選択することができます。ホットキーの **S** を押してアクセスできます。

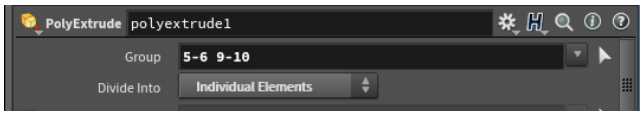
Secure Selection - 他のツールを使用する間、現在の選択をロックします。これがオンの状態で **Select** ツールを呼び出すには、ホットキー **S** を押したままにします。

選択タイプ - 上部のバーにあるこのオプションを使用して、選択のタイプを変更できます。Box、Lasso、Brush、Laser のいずれかを選択できます。フィルタオプションもいくつかあります。

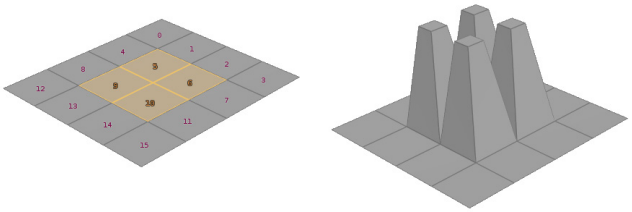
エッジループ - エッジループを選択するには、エッジを選択した状態で**ダブルクリック**します。不完全なループを選択するには、一方の端を選択して **A** を押してから、もう一方の端のエッジを選択します。これは、ポイントやプリミティブでも機能します。同じテクニックを使用して、ポイントループやプリミティブループも選択することができます。

選択コンポーネントのツールでの扱い

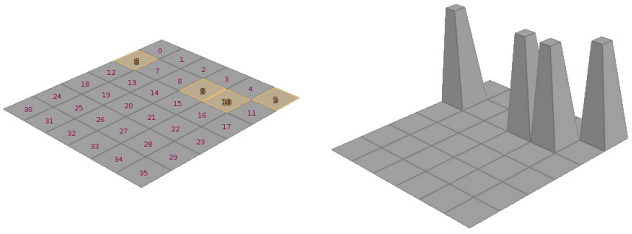
ビューポートでポイント、プリミティブ、またはエッジを選択してからツールを使用すると、ノードが作成され、選択がノードの **Group** パラメータにリストされます。



例えば、ここではプリミティブ **5、6、9、10** が ***polyextrude*** ノードで使用されていることが分かります。Group ノードにリストされ、フェースを押し出すのに使用されているのを確認できます。



入力ジオメトリノードのトポロジを変更した場合、フェースの数が増減したり、押し出しの場所が移動することがあります。このような場合は、必要に応じてフェースを再選択します。



これを行うには、***polyextrude*** を選択し、**Enter** を押して **Handle** ツールに移動してから、` を押して再選択モードに切り替えます。新しいプリミティブを選択してから **Enter** を押すと、新しい選択が **Group** パラメータで使用されます。

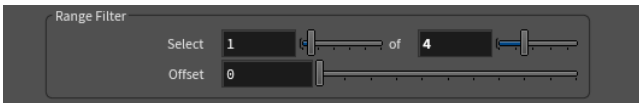
すべて選択と Group フィールド

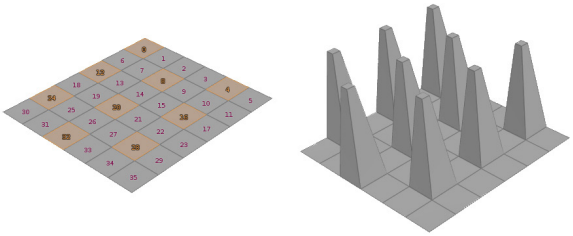
入力ジオメトリのすべてのプリミティブを選択するには、**Group** パラメータを空のままにします。入力ジオメトリのトポロジが変更された場合も、ノードがすべて処理してくれます。

ビューポートで**すべて選択 (N)** を使用すると、ツールの使用時、通常このフィールドは空のままになります。一部のツールでは、Group フィールドに選択したすべてのパーツが表示されるので、手動でフィールドをクリアして空にしなければなりません。

Group ノード

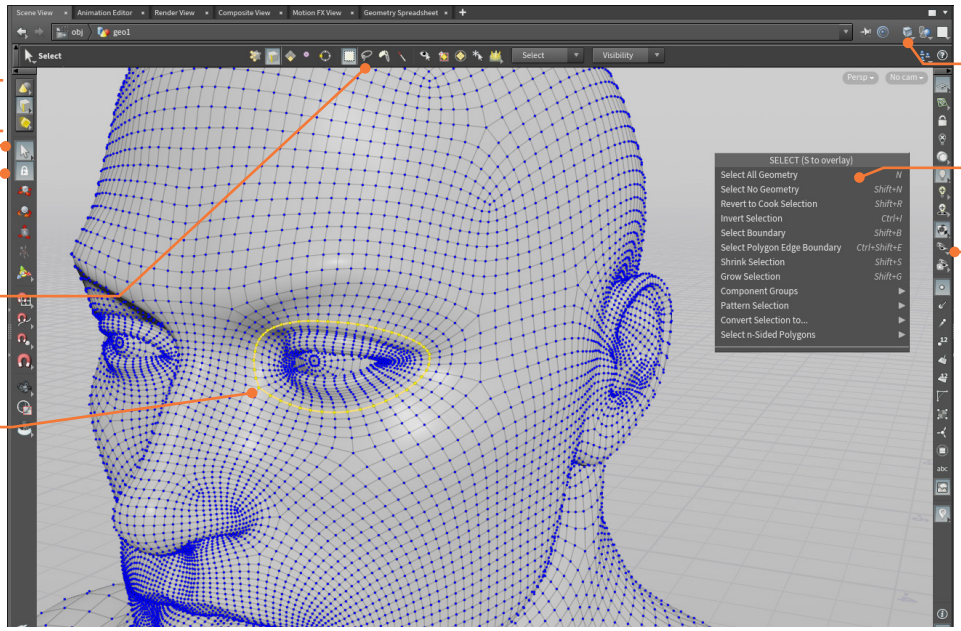
Group ノードを使用すると、ポイント、頂点、ポリゴン、エッジの選択を**名前**で定義し、参照することができます。グループの定義は、ビューアでコンポーネントを選択するか、範囲やエクスプレッションを使って数学的に行います。その後、ポイント番号やプリミティブ番号を使う代わりに、**Group** パラメータにグループ名を割り当てます。





使用可能な **Group** ノードをいくつか紹介します。

- **Group Create** - インタラクティブな選択、境界ボックス、フェースの法線の向き、エッジの角度を使用して、グループを作成します。
- **Group by Range** - 範囲とシンプルなパターンを選択して、グループを作成します。
- **Group Expression** - VEX エクスプレッションを使用して、グループメンバーシップを定義します。
- **Group Paint** - インタラクティブなペイントインターフェースを使用して、グループ化するジオメトリを選択します。



シェーディングオプション - Scene View に何を表示するかを決定します。ここでは Smooth Wire Shaded が選択されています。

RMB メニュー - Select ツールの使用時、このメニューでは選択の反転、境界の選択、選択の拡大と縮小などの選択オプションにアクセスできます。

ディスプレイフィルタ - ボーン、Null オブジェクト、ライト、カメラなど、必要ないものを非表示にして、作業に集中できるようにすることができます。

Display Options - 選択モードは、選択しやすいようにエッジやポイントを示してくれますが、他のツールの使用時、それらは表示されません。これらのオプションを使用すると、特定のモデリングツールを使用していないときでも、さまざまな要素を表示したままにすることができます。

トランスフォームと編集

オブジェクトの基本のトランスフォームツールから、アニメーションリグの Pose ツール、ジオメトリを形状変更する Edit ノードまで、ビューポートでインタラクティブなハンドルを使用できる各種ツールが用意されています。Houdini では、これらのハンドルは、作業中のノードと密接に結びついています。

トランスフォームツール

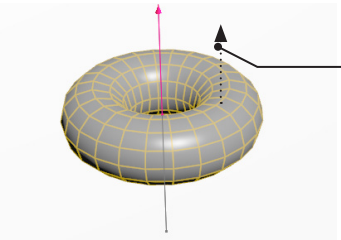
トランスフォームツールでは、ハンドルを使用してオブジェクトを操作したり、ジオメトリの形状を変更します。オブジェクトをトランスフォームすると、オブジェクトレベルのパラメータが更新され、変更内容が反映されます。

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

Handle ツールを使用すると、選択したノード特有のハンドルが表示されます。これらのツールを使用している間は、**S を押したままにする**ことで再選択できます。新しく選択したら **S を放して**、トランスフォームを続けます。

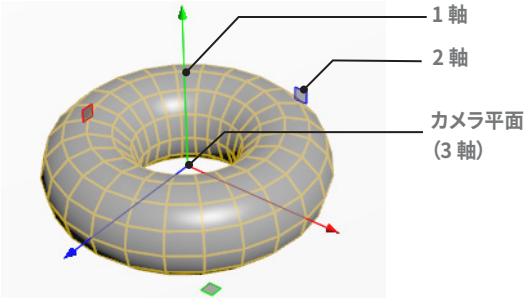
MMB 移動

ハンドルを直接クリックしたくない場合は、空きスペースで**中マウスボタン**をクリックしながらドラッグすると、コンストラクション平面に沿って移動できます。最も近い軸に沿って移動するように変更するには、**Edit > Preferences > Handles** で Translate Handle を **Map Drag to Axis** に設定します。



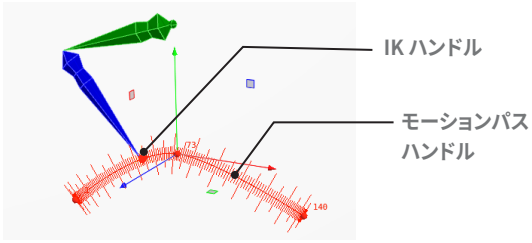
トランスフォームハンドル

Move ハンドルを使用すると、1 軸または 2 軸で移動したり、中心を使用してカメラ平面に沿って移動することができます。**Rotate** ハンドルと **Scale** ハンドルでも同様の操作が可能です。



POSE ツール

アニメートするときは、**Pose** ツールを使用してボーンを操作したり、オブジェクトの動きを示すモーションパスハンドルを表示します。その後、接線ハンドルやキーフレームを使用して、ビューポートで動きを変更します。



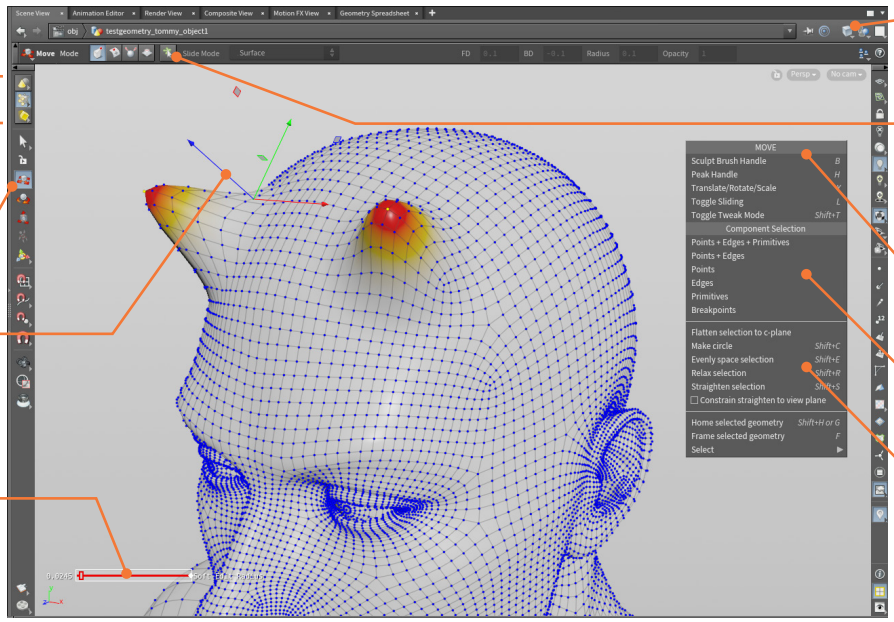
ジオメトリの編集

編集 | コンポーネント - これらのボタンを使用して、編集したいコンポーネントを選択します。ここでは **Points** オプションが選択されています。

Move ツール - **Move** ツールでは、Scene View のハンドルを使用して選択を移動できます。

Move ハンドル - 矢印を使用すると 1 軸で移動でき、正方形のドットを使用すると 2 軸で移動することができます。ハンドルを **RMB クリック**すると、ハンドルオプションが表示されます。

Soft Edit Radius - サーフェス上のポイントを移動するとき、この半径値を使用して、ソフトな減衰を作成できます。プリミティブまたはエッジでは、Soft Edit Radius は機能しません。



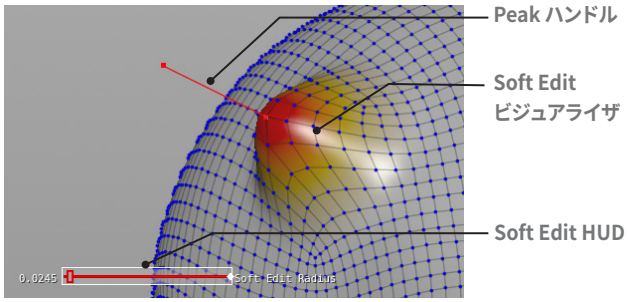
Edit ノード

ジオメトリコンポーネントを移動しようとすると、**Edit ノード**が配置され、トランスフォームを受け入れます。ジオメトリのトランスフォームだけでなく、サーフェス上を滑るようにしたり、法線に対して垂直にトランスフォームしたり、Sculpt を適用することも可能です。

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

ソフトな減衰

ポイントをトランスフォームするとき、**Soft Edit Radius** 使用すると減衰を作成できます。ビジュアライザが呼び出され、サーフェス上のどこで減衰が発生しているかを確認できます。



EDIT オプション

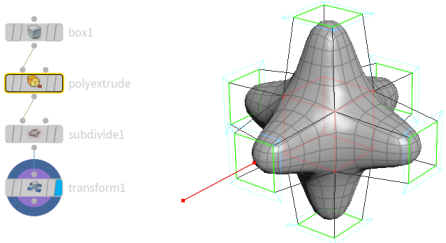
Edit ノードを **RMB クリック**すると、選択をトランスフォームするためのオプションが表示されます。選択を円にしたり、まっすぐにすることができます。これらのオプションは、ポイントとエッジに対して機能しますが、プリミティブに対しては機能しない場合もあります。

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

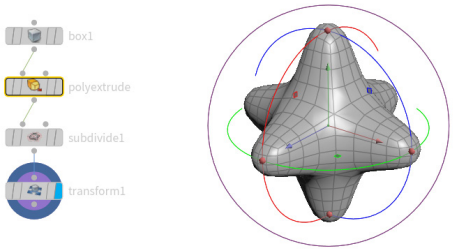
Handle ツール

シェルフツールを使った後は、たいてい **Handle** ツールに切り替えられています。また、ネットワーク内のノードを選択し、Scene View で **Enter** キーを押しても、**Handle** ツールに切り替わります。これにより、**polyextrude** ノードの ***distance*** パラメータなど、選択したノード特有のパラメータにフォーカスしたハンドルが表示されます。

Show Current Operator - デフォルトでは、表示ノード以外のノードを選択すると、それが現行ノードとなり、ジオメトリのワイヤーフレームが表示されます。その後、ハンドルを使用してこの中間ノードを操作しながら、シェーディングサーフェス上で結果を評価できます。



Show Display Operator - もう 1 つのオプションは、表示オペレータを常に表示します。この場合、チェーン内のノードを選択してもワイヤーフレームは表示されず、ハンドルは表示ノードにフォーカスしたままになります。



パラメータは現行ノードのパラメータエディタで変更できますが、ハンドルは表示ノードのパラメータで機能し続けます。

シェーディングオプション - Scene View に何を表示するかを決定します。ここでは、Smooth Wire Shaded が選択されています。

Sloppy Selection - Edit ノードが Sloppy 選択を使用している場合、3 つのコンポーネントボタンを同時に選択することができます。これらすべてのボタンを使用したより流動的な選択プロセスが可能です。

RMB メニュー - フォーカスしたい編集のタイプなど、Edit ツールオプションにアクセスできます。Scene View の上部のバーでもこの情報を利用可能です。

コンポーネントの選択 - このメニューを使用してコンポーネントタイプを選択できます。メインツールバーにもこれと同じオプションがあります。

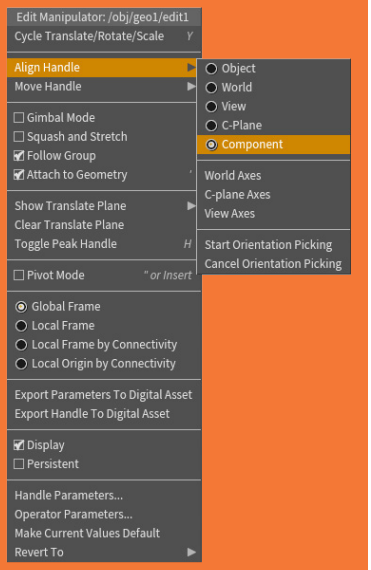
Edit オプション - **Make Circle** や **Straighten selection** などのオペレーションを使用して、コンポーネントを編集できます。

ハンドルオプション

すべてのハンドルにメニューがあり、任意の部分を **RMB クリック**することでアクセスできます。

このメニューには、ハンドルを整列したり、ノードのパラメータから切り離したり、Pivot Mode に設定するなどのオプションが用意されています。これらのオプションを使用して、ハンドルの動作をカスタマイズできます。

また、ハンドルのパラメータすべてをキーフレームに設定したり、ハンドルの全パーツをデジタルアセットにプロモートすることもできます。パラメータをプロモートすると、アセットレベルでハンドルにアクセスできるようになります。



モデリングツール

Houdini には、ジオメトリを作成、形成、変形して目的の外観にするツールが多数用意されています。ここでは、Houdini のジオメトリ（SOP）コンテキストでモデルを構築する際、よく使用する各種ツールのうち一部を取り上げます。

作成

ジオメトリの作成は、基本形状を使用するか、カーブを描画することから開始します。いずれの場合も、そのツール名のオブジェクトが作成され、その内部にジオメトリ / SOP ノードが含まれます。ツールには **Create** シェルフまたは Radial メニューからアクセスできます。

プリミティブ - Houdini には、Box、Sphere、Tube、Torus プリミティブ形状と、さまざまなプラトン立体があります。

Grid - Grid ツールは、多様なモデルの出発点として最適です。ジオメトリレベルで形状とサイズを設定できます。

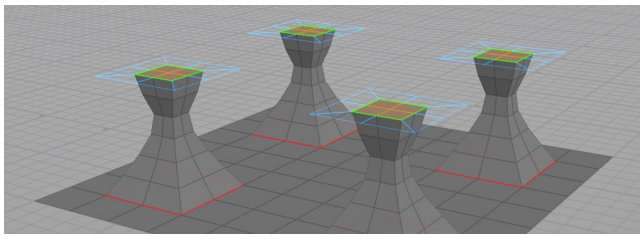
Curve - コントロールポイントを配置しながらカーブを描画し、その後 Bezier、NURBS、またはポリゴンカーブを作成します。

ポリゴンモデリング

ポリゴンは、最も一般的なジオメトリタイプの 1 つで、中でもビデオゲームプロジェクトでは必須です。Houdini には、モデルの開発に使用できる、包括的なポリモデリングツールセットが含まれています。

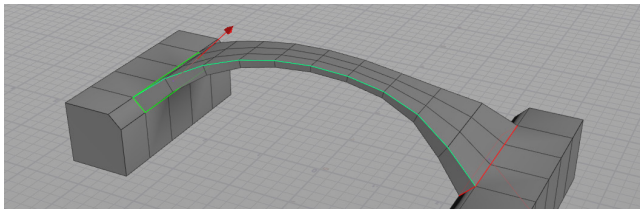
PolyDraw - このツールを使用すると、コンストラクション平面で、または既存のジオメトリにスナップすることで、インタラクティブにポリゴンメッシュを描画できます。

PolyExtrude - 1 つまたは複数のポリゴンを押したり引いたりして、ジオメトリの形状を変更します。押し出しのプロファイルを制御して、さまざまな形状を得られます。



PolyBevel - 選択したエッジにベベルをかけて、まっすぐな面取りや丸い面取りを作成します。通常、Polyxtrude や Boolean などの前のノードの出力グループを使用して、適切なエッジを自動的に見つけられます。

PolyBridge - ブリッジの形状を制御しながら、2 組のポリゴンをつなげます。



PolySplit/Edge Loop/Knife - これらのツールを使用すると、ポリゴンを分割してモデルにディテールを追加できます。

PolyExpand 2D - 2D 平面上のカーブやエッジを取り込み、任意のオフセット値に基づいてジオメトリを作成します。

PolyReduce - 四角形トポロジと UV を維持したままポリゴン数を減らすことで、異なるレベルのディテールを作成できます。

PointWeld - ポイントのグループを他のターゲットポイントにインタラクティブにスナップさせ、それらを結合します。

ユーティリティノード

Houdini はプロシージャルなため、コピー、クリップ、ミラーなどのモデリングアクションを行うと、ネットワークにノードが作成されます。この仕組みにより、後で戻って変更を加えるのも簡単にできます。

Clip - クリッピング平面を基準にモデルを切断します。クリップの方向を設定したり、片側、もう片側、または両方を保持するかどうかを選択できます。

Mirror - クリッピング平面を基準にジオメトリを反転します。ミラー後にポイントを結合するオプションがあります。

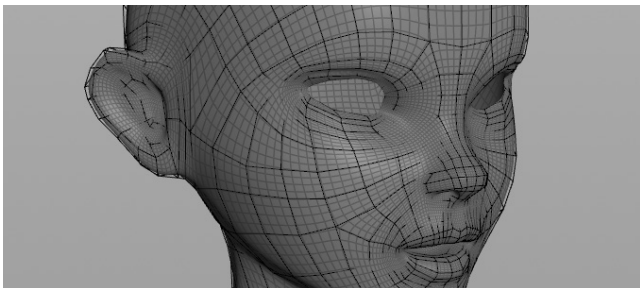
Copy and Transform - トランスフォーム値に基づいて複数のコピーを作成します。

Blast - モデルからポリゴンを削除します。選択したポリゴンを削除するか保持するかを選択できます。ポイントまたはポリゴンを選択している状態で **Delete** キーを押すと、それらを削除できます。

Dissolve - 周囲のジオメトリを壊すことなく、エッジを削除できます。エッジを選択した状態で **Delete** キーを押すと、そのエッジを削除できます。

サブディビジョンサーフェスモデリング

Houdini では、ポリゴンでモデリングした後、オブジェクトのパラメータエディタの **Render** タブにあるオプションを使用して、サブディビジョンサーフェスとして**表示**したり**レンダリング**することができます。また、ジオメトリレベルで **Subdivide** ノードを作成し、ポリゴンを追加すれば、よりディテールの多いトポロジで作業できます。



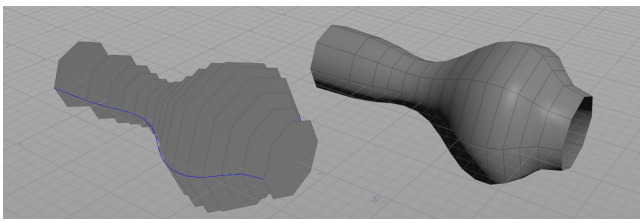
サーフェス化ツール

Houdini には、プロファイルカーブを受け取ってサーフェスを構築するツールがあります。入力カーブは、Bezier、ポリゴン、NURBS カーブのいずれか、またはそれらの混合です。

Revolve - 軸を基準にプロファイルカーブを回転させて、ジオメトリを作成します。結果はハンドルを使って微調整できます。

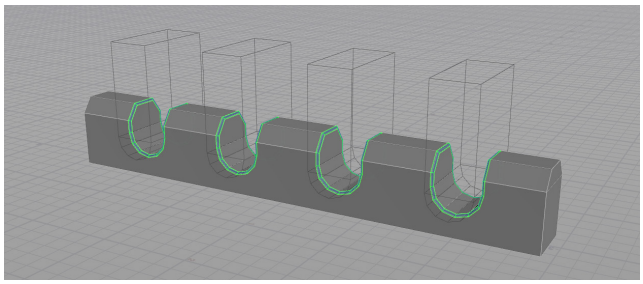
Skin - 複数のカーブを受け取り、それらをサーフェスに変換します。

Rails - 1 つまたは複数のプロファイルカーブを 1 つまたは複数のレールカーブに沿ってコピーしてから、結果をスキン化してサーフェスを取得します。



BOOLEAN

Boolean ツールを使用すると、ジオメトリの**減算(Subtract)**、**結合(Union)**、**交差(Intersect)**を計算できます。このノードは非常に複雑なトポロジを扱うことができ、サーフェスを分解して、リジッドボディダイナミクスを用いた破壊を作成することができます。多くの場合、ボロノイベースの **Shatter** ノードよりもリアルな結果を得られます。

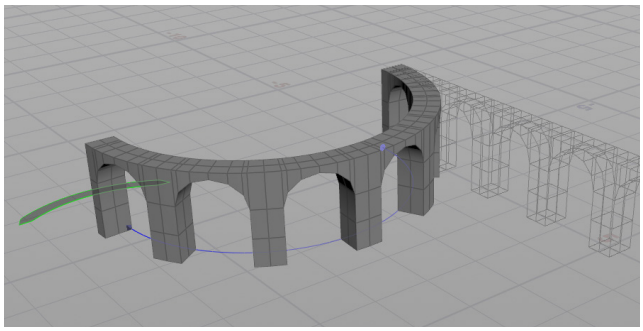


Boolean ツールは出力グループを作成し、それを使って **Polybevel** ノードなどの他のノードに接続できます。こうすることで、Boolean に対する更新は、2 つ目のノードに渡されたタイミングで適切に更新されます。

変形ツール

ポイントを直接編集してジオメトリを形づくることもできますが、より汎用的なアプローチが必要な場合もあります。次のノードは、ジオメトリをプロシージャルに形成するためのオプションを備えています。

Bend - キャプチャ範囲と方向を設定して、含まれるジオメトリに対して曲げ、捻じり、テーパ、収縮などを適用します。



Lattice - ジオメトリの周りにラティスを構築し、ケージ上のポイントを編集してジオメトリの形状を変更できるようにします。カスタムのケージも使用可能です。

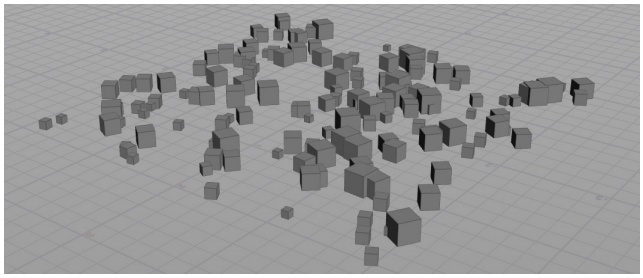
Mountain - ノイズ関数を適用してサーフェスを変形し、ランダムな結果を作成します。このノードでは、ポイントが実際に動かされます。

Ripple - ジオメトリに波紋の形状を作成します。

Waves - ノイズ関数を追加して、時間の経過に応じてアニメートされる、波のようなパターンを作成します。リアルな海を作成するのに最適です。

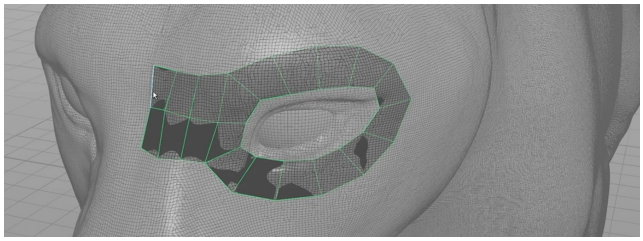
COPY TO POINTS + SCATTER AND ALIGN

一般な Houdini ワークフローでは、サーフェス上にポイントをばら撒いて整列させてから(**Scatter and Align**)、ポイントにコピー(**Copy to Points**)します。その後、オブジェクトをスケールしたり回転するためのアトリビュートを適用すると、有機的な結果が得られます。これは、木や岩を含む風景を作成する際によく使用される方法です。



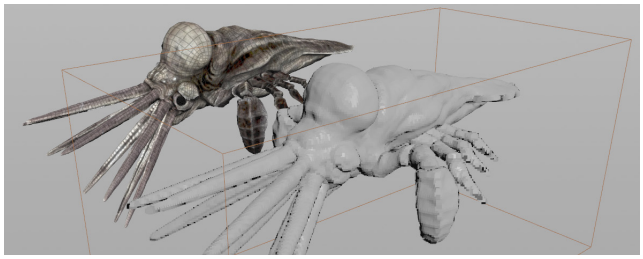
TOPOBUILD

Houdini の **TopoBuild** ノードを使用すると、Pixologic の ZBrush ようなアプリケーションでスキャンしたり作成した高解像度のジオメトリに直接ポリゴンを描画できます。アニメーション用のクリーンなトポロジを作成してから、元のモデルのディテールを法線マップにベイクできます。



ボリューム

ボリュームには、空間内のボクセル(3 次元ピクセル)の値を格納できます。ダイナミクスツールを使用する際の衝突をサポートしたり、雲を作成するためによく使われます。また、モデリングでは、複数の形状を 1 つのボリュームにまとめ、それを変換してサーフェスに戻すことも可能です。



ジオメトリタイプ

Houdini は、**プリミティブ**、**ポリゴン**、**NURBS**、**Bezier** など、さまざまなジオメトリタイプをサポートしています。こうしたタイプ間の**変換**が可能であり、1 つのオブジェクト内で複数のジオメトリタイプを結合することもできます。

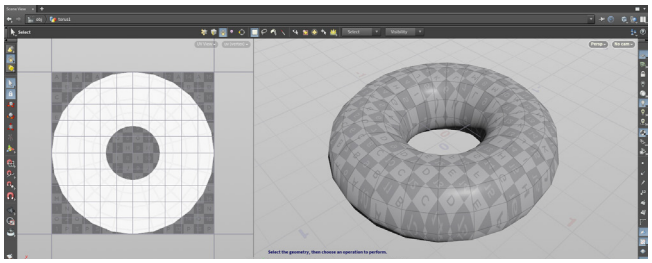
ポリゴンモデルは、**Pixar の OpenSubdiv** 標準を使用して、**サブディビジョンサーフェス**として表示およびレンダリングするようセットアップすることができます。サブディビジョンと NURBS は両方とも、テッセレーション設定に依存することなく、Karma と Mantra で非常に滑らかにレンダリングされます。

UV とテクスチャ

2D マップを 3D オブジェクトに適切にフィットさせるには、ジオメトリの平坦化したビューを定義する UV 座標が必要です。Houdini で最初にジオメトリを作成したときには、UV はありません。プリミティブオブジェクトにも、組み込みの UV はありません。これは、1 つまたは複数の SOP ノードを使用して、ジオメトリレベルで UV を追加する必要があることを意味します。

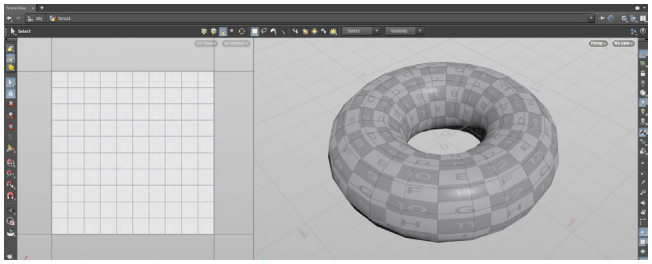
UV テクスチャの表示

Houdini のジオメトリには、デフォルトでは UV テクスチャアトリビュートがセットアップされていないため、UV ツールを使用して追加する必要があります。UV をセットアップすると、**Display Options** バーの **Show UV Texture** がオンになり、ジオメトリにテクスチャグリッドが表示されます。UV テクスチャを表示したくない場合や、カラーテクスチャに変更したい場合は、オフにします。



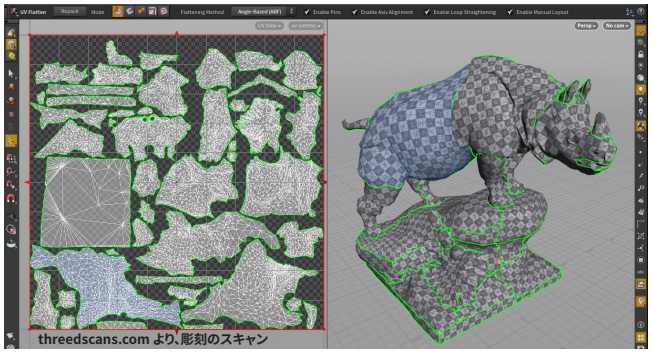
UV PROJECT

このノードでは、いくつかある投影テクニックの 1 つを使用して、UV を割り当てます。投影タイプを選択したら、オブジェクトに合わせて投影を**初期化**します。これにより、UV が反転される場合があるため、**Rotate X** の値を 90 ではなく **-90** に設定する必要があります。上の図は **Orthographic**（正投影）、下の図は **Toroidal**（ドーナツ状）投影です。



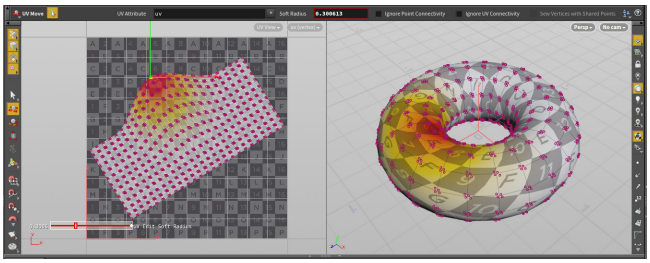
UV FLATTEN

UV Flatten は、選択したエッジまたはエッジグループで事前定義した境界を基に、ジオメトリをアンラップ（展開）します。**UV ビュー**でポイントをピン留めしたり、希望するルックになるよう島を調整して、結果を微調整できます。



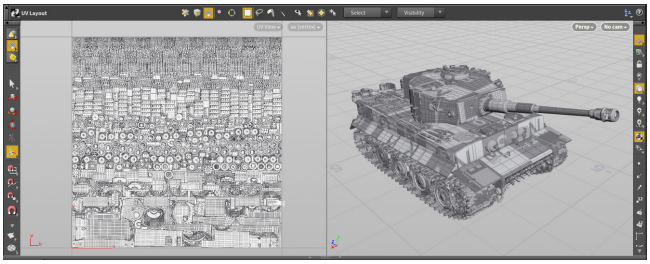
UV Edit と UV Distortion

個々の頂点や頂点グループを編集するには、**UV Edit** または **UV Transform** ノードのいずれかを使用します。UV Edit ノードは 1 つのノードで大量の編集を行えますが、UV Transform は 1 つのノードで行える編集は 1 つです。そのため、よりプロシージャルな結果を得られます。**UV ビューポートメニュー**で **Display > UV Distortion** を選択すると、どの程度多くの編集を加えたかを確認できます。



UV LAYOUT

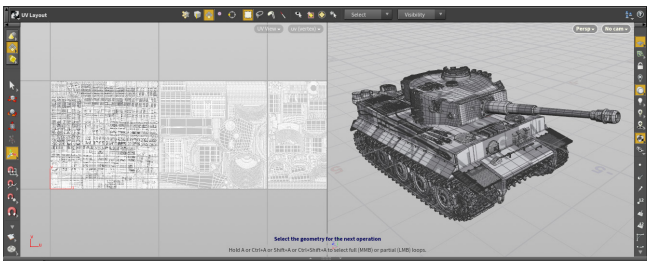
UV Layout は、UV 島を作成し、可能な限り効率的に UV 空間内に詰め込みます。ジオメトリ上で利用されるテクスチャ量が最大化され、レンダリングとゲームプレイの両方を最適化できます。



領域ハンドルを使用して、UV レイアウトを UV 空間の特定の部分に配置できます。後続のレイアウトがこのレイアウトを避けるようにするには、**Pack Island in Cavities of Other Islands** オプションを使用します。

UDIM

使用できるのは単一の UV タイルだけではありません。UDIM を使用すると、UV を多数のタイル上に拡張できます。このテクニックでは、UV 島が密接に詰め込まれないため、ディテールの豊富なテクスチャマップを作成できます。テクスチャマップに適切に番号が付けられ、適切なタイルに割り当てられます。



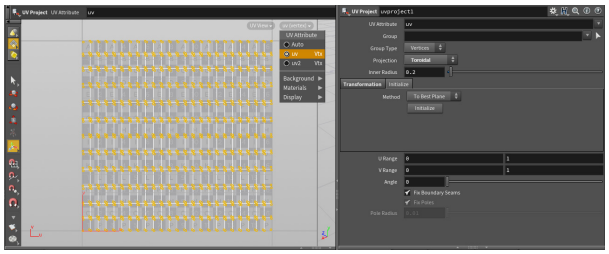
UV アトリビュート

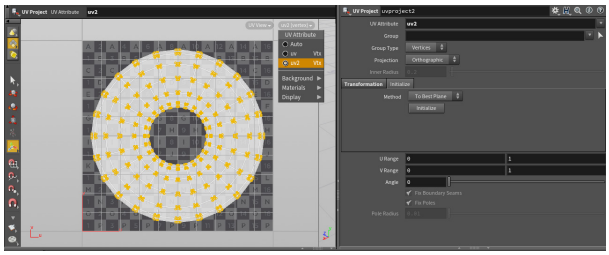
前の項では、ジオメトリにアトリビュートを割り当てて、重要な情報をネットワークに伝えられるのだと学びました。UV は、テクスチャマップをモデルにラップできる頂点アトリビュートで、同じようにネットワークに伝達されます。

これらのアトリビュートは UV ビューポートで視覚化され、Geometry Spreadsheet で分析されます。このようなアトリビュートは、上級 TD がスクリプトを使用して UV を管理するための Attribute Wrangle ノードをはじめ、さまざまな SOP ノードで使用できます。

UV セット

同一ジオメトリに複数の UV セットを作成できます。UV ノードを使用するときに、UV アトリビュートを設定します。デフォルトでこれは ***uv*** ですが、***uv2*** を作成して、2 つ目のセットを作成することも可能です。異なる UV セットは、VOP でテクスチャを割り当てるときに使用します。これにより、テクスチャマップごとに異なる UV アトリビュートを使用できます。

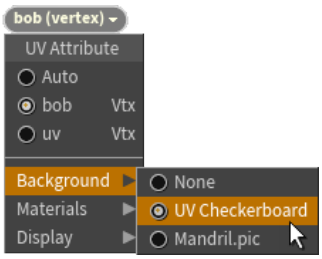




上の 2 つの画像では、1 つ目は Toroidal（ドーナツ状）投影で ***uv*** **UV アトリビュート**に割り当てられていますが、2 つ目は Orthographic（正投影）で***uv2*** **UV アトリビュート**に割り当てられています。これらの UV アトリビュートには任意の名前を付けることができ、例えば ***uv2*** ではなく ***bob*** としてもかまいません。

UV ビューポートメニュー

UV ビューポートメニューを使用すると、UV アトリビュートに基づいて UV を表示できます。また、このメニューでは背景画像を確認することもできます。背景画像は、デフォルトの UV グリッドか、割り当てられたマテリアルから引き出されたテクスチャマップです。

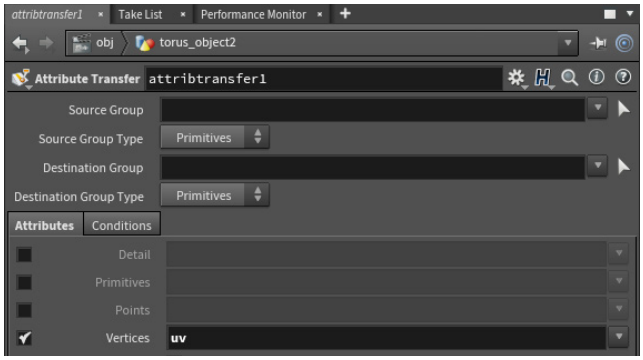


このメニューには、**UV Overlap**、**UV Backfaces**、**UV Distortion** といったディスプレイオプションもあります。これらのオプションは、UV を評価して、さらに微調整が必要かどうかを判断するのに役立ちます。

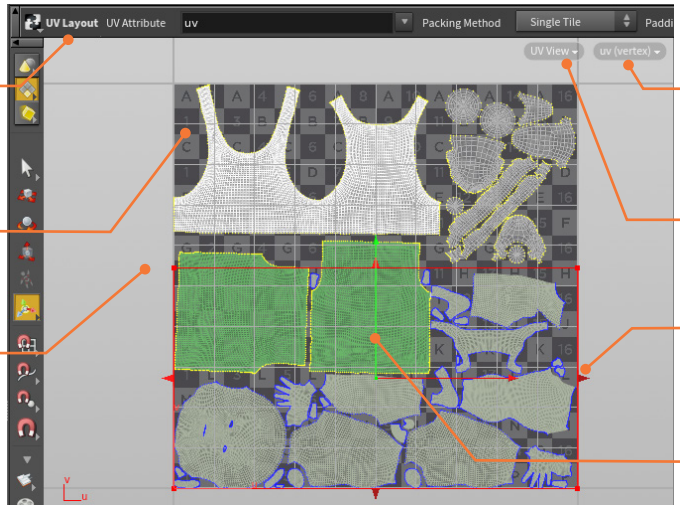
ATTRIBUTE TRANSFER

アトリビュートの管理に使用できる SOP ノードの 1 つである **Attribute Transfer** を使用すると、あるジオメトリの UV アトリビュートを、Proximitity（近接度）に基づいて別のジオメトリに転送することができます。

これは、モデルのトポロジを変更したが、元のモデル用に UV を作成した際の作業内容を保持したい場合に便利です。



SCENE VIEW | UV



現行ツール - Scene View の上部には、Handle ツールがアクティブであれば、選択したノードが表示されます。

背景 - メインタイルの背景は、UV メニューのオプションを使用して設定できます。

メインタイルの外側 - 複数のタイルをカバーする UDIM を使用しない限り、テクスチャが繰り返されるため、メインタイルの外側の領域に配置されたポリゴンはメインタイル上の同じ領域とオーバーラップします。

UV メニュー - UV ビューにいるとき、このメニューには UV に関連したさまざまなオプションが表示されます。

ビューメニュー - このメニューでは、このビューポートに表示するの UV ビューを選択できます。

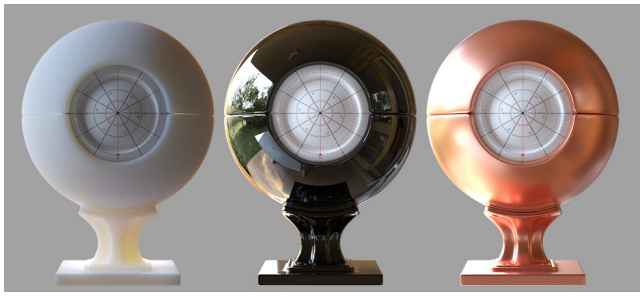
レイアウトハンドル - このハンドルは、UV Layout ノードの一部です。タイルの特定の部分の UV に集中できます。

既存データを基準とした詰め込み - UV が既に設定されている既存のジオメトリの周囲に、UV を詰め込みます。

ルックデブ：シェーダとマテリアル

シーン内のオブジェクトをレンダリングするには、シェーダとも呼ばれるマテリアルをジオメトリに割り当てる必要があります。Houdini では、これらのマテリアルやシェーダは Material/Vex Builder ネットワークで作成されます。ノードを使用してマテリアルを構築する機能は、ショットのルックを定義する際の強力なツールとなります。

Houdini では、さまざまなタイプのノードをいくつかのネットワークタイプに分けていますが、マテリアルの場合は `/mat` ネットワークタイプを使用します。ここでは、Karma と Mantra 向けの **VEX オペレータ**や、Karma 向けの **Material X** をセットアップすることができます。MaterialX は、Lucasfilm® が開発したオープン規格で、アプリケーションとレンダラ間でルックデベロップメントコンテンツを転送するために使用されます。



MATERIAL PALETTE

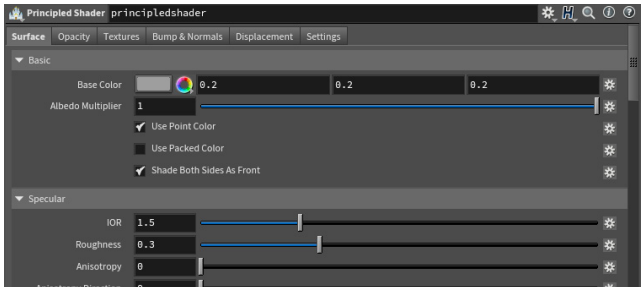
Material Palette を使用して VEX ベースのマテリアルを追加し、クリックアンドドラッグでオブジェクトに割り当てることができます。このペインには、シーン内のマテリアルを管理するワークスペースがあり、**Material Library LOP** など、サブネットワークを表すタブに分かれています。**Tab キー**を使用して、MaterialX シェーダをマテリアルネットワークに追加できます。配置すると、Material Palette に表示されるようになります。

LOP でのマテリアルの割り当て

Solaris LOP コンテキストでマテリアルを割り当てるには、まず ***/mat*** ネットワークを含む **Material Library LOP** を作成します。このノードを使用するか、下流で **Assign Material LOP** を使用して、マテリアルを割り当てられます。**Material Palette** では、ギャラリーからライブラリにマテリアルをドラッグできます。LOP では、**矢印ボタン**を使ってシーングラフのマテリアルリストにアクセスできます。

Principled Shader

Material Palette には、Brent Burley 氏による Disney "principled" BRDF（Disney 原則 BRDF）をベースとしたマテリアル、Principled Shader が含まれています。このシェーダは、アーティストが扱いやすいように、物理的ではなく「原則に基づいた」ものになっています。**Principled Shader** は、Base Color、Bump、Normal、Displacement などのパラメータに直接テクスチャを割り当てられるようになっています。割り当てたテクスチャマップはビューポートに表示され、簡単にさまざまなルックを実現することができます。
このマテリアルは他の VOP に接続することで拡張可能ですが、それが必須なわけではありません。ギャラリーに含まれるマテリアルの多くは、このシェーダのバリエーションです。



PRINCIPLED SHADER CORE

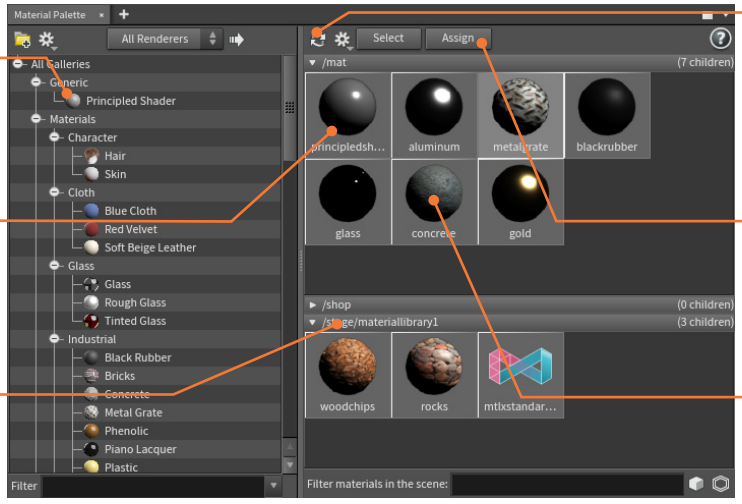
Principled Shader Core ノードは、**Principled Shader** 内にあり、シェーディングモデルの主な機能を含んでいます。ただし、すべてのテクスチャリング機能を搭載しているわけではありません。
このノードを使用してゼロから堅牢なシェーダを構築するには、Houdini のシェーダ構築ツールを使用して VOP ノードを追加する必要があります。このためには、ノードビューでノードを接続したり、Shader FX メニューを使用してノードを追加したりします。

MATERIAL PALETTE

ギャラリー内のマテリアル - ここにリストされるマテリアルは、ディスク上のギャラリーファイルに保存されています。右側のシーン領域か、ビューポートのオブジェクトにドラッグできます。

シーン内のマテリアル - ここにはシーンファイルの一部であるマテリアルが表示されます。ここからビューポートにドラッグすることで、オブジェクトに割り当てることができます。

Material Library LOP - LOP コンテキストでセットアップされたマテリアルは、Material Library に配置できます。ギャラリーからここにマテリアルをドラッグすることができます。



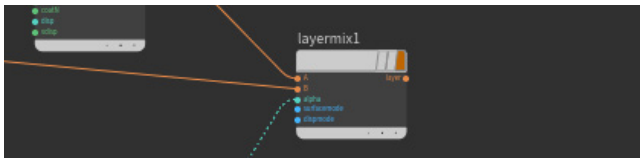
Update Material アイコン - すべてのマテリアルアイコンを更新するには、このボタンをクリックします。1 つずつ更新するには、マテリアルを RMB クリックします。

マテリアルの割り当て - シーン内のオブジェクトとパレット内のマテリアルを選択すると、このボタンを使用してマテリアルを割り当てることができます。

ダブルクリックして編集 - いずれかのマテリアルをダブルクリックすると、ノードビューにジャンプして、/mat レベルで編集できるようになります。

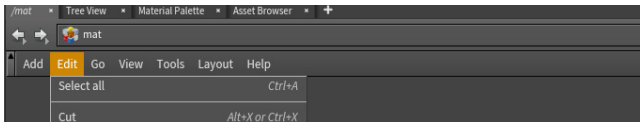
マテリアルのレイヤー化

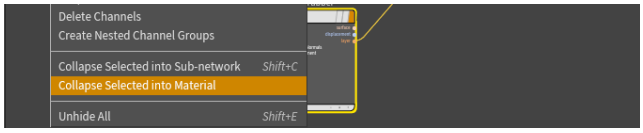
オブジェクトをユニークなルックにするために、マテリアルをレイヤー化することができます。**Layer Mix** ノードを使用して 2 つの異なるマテリアルを組み合わせ、1 つのルックを作成します。例えば、このテクニックを使って光沢のある金属マテリアルとマットな錆マテリアルをレイヤー化することが可能です。そうしておいてから、アルファチャンネルをテクスチャ化し、サーフェスあるいはディスプレイスメント、またはその両方を組み合わせたマテリアルを設定します。



MATERIAL BUILDER

レイヤー化したノードを新しいマテリアルに変換したい場合は、ノードを選択して、**Edit > Collapse Selected into Material** を選択します。これにより、ノードが **Material Builder** 内に配置され、引き続き微調整を行えます。このレベルでは、***output*** と ***collect*** ノードがあり、ネットワークを効率的に動作させることができます。



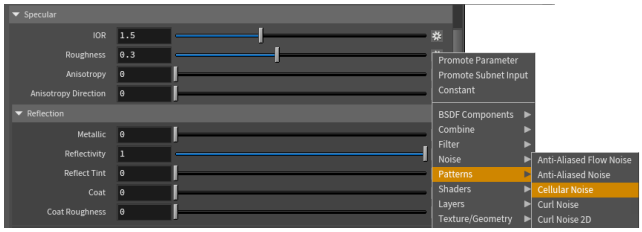


デジタルアセットとしてのマテリアル

マテリアルを **Houdini デジタルアセット**として保存することで、マテリアルをさらに効率化できます。**Asset Properties** ペインで、**Save** タブに移動して **Save Cached Code** を選択すると、Mantra でレンダリングするときにマテリアルが事前にコンパイルされます。また、テクスチャマップをデジタルアセットに読み込んで、アセットファイル内からそれらを参照することも可能です。マテリアルを HDA に変換することで、チームとの共有が容易になります。

SHADER FX メニュー

マテリアル VOP を使用するときは、ネットワークビューでノードを追加して接続するか、各パラメータの右端にあるアイコンをクリックすると表示される **Shader FX メニュー**を使用します。このメニューを使用すると、作業したいパラメータにフォーカスして、コンテキストでノードを作成できます。



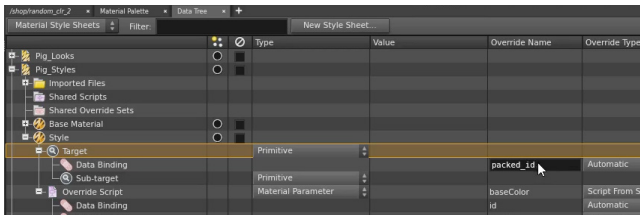
パラメータエディタで、右端のアイコンから各パラメータの接続タイプを確認できます。

- 接続なし
- Parameter ノード
- 他のノードと接続
- 非表示の接続

プロシージャルなマテリアルの割り当て

多くのデータを扱うプロダクションでは、多くの場合、プロシージャルなアプローチでマテリアルを割り当てる必要があります。**Solaris** や **Karma** でこれを実現するには、**Assign Material LOP** や **Material Variation LOP** などのノードを使用します。

Mantra を使用している場合、**Data Tree** パネルを使用してマテリアルをオブジェクトに割り当てることができ、このペインから **Material Stylesheets** にアクセスできます。スタイルシートでは、ルールを使用して大規模なオブジェクトグループにマテリアルやテクスチャを割り当てられます。

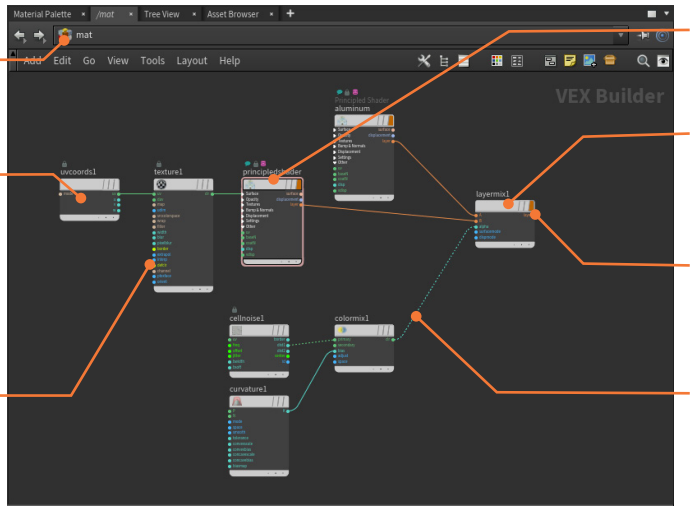


シェーダ構築

ノードパス - 現在のパスを確認でき、マテリアルネットワーク内を移動しやすくなります。

VOP ノード - マテリアルコンテキストでは、マテリアルノードから開始し、VOP ノードを接続して、マテリアルのテクスチャリングをカスタマイズします。完了したら、すべてを折り畳んで Material Builder ノードに戻すことができます。

ノードコネクタ - ドットを **MMB クリック**してアクセスする Shader FX メニューを使用して、この領域にノードを追加できます。ドットを **RMB クリック**すると、完全なノードメニューが表示されます。



Principled Material - 単体で割り当てたり、Layer Mix に接続できる代表的なマテリアルです。

Layer Mix - マテリアルでのレイヤー出力は、このミックスノードに送ることができます。これをジオメトリに割り当てられます。

Material フラグ - レイヤーのミックスを Material Palette に表示させたい場合は、このフラグをオンにします。

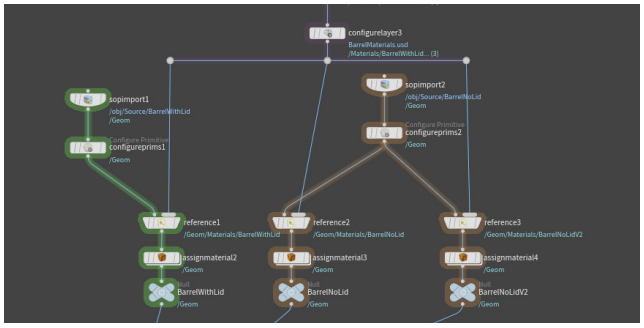
アルファ - ここでは、VOP ノードが Layer Mix ノードのアルファを送り、2 つのレイヤーマテリアルのアルファマスクを作成しています。

Solaris : レイアウト

Solaris は、ルックデブ、レイアウト、ライティングに特化した Houdini のコンテキストで、USD が中核にあります。このコンテキストに取り込まれたオブジェクトやジオメトリは USD となり、専用のノードを使用してオブジェクトの配置、ジオメトリのインスタンス化、ショットレイアウトの管理などを行えます。

SOLARIS : LOP

Solaris 環境を使用するには、`/stage` ネットワークに移動するか、LOP ネットワークを作成します。ここに、ジオメトリの取り込み、マテリアルの割り当て、ライトやカメラの追加を行うためのノードが配置されます。これらのノードでは、共有アセットを使用してシーケンスやショットを作成したり、プロシージャルな編集ノードで各ショットの設定をカスタマイズすることができます。



その中核において、Solaris 環境はすべてを USD（**Universal Scene Description**）に変換します。USD は、PIXAR によって作成されたオープンソースイニシアティブです。Solaris/LOP コンテキストでは、プロシージャルノードで USD をネイティブに使用し、参照、ペイロード、レイヤー、コレクション、バリアント、詳細レベルといった USD の概念を管理できます。

LOP ネットワークは、Karma または他の Hydra 互換レンダラを使用してレンダリングできます。Hydra は、USD をビューポートにレンダリングしてインタラクティブに探求したり、最終レンダリングとしてディスク上にレンダリングするためのテクノロジです。

SCENE IMPORT : オブジェクトを LOP に

Houdini のオブジェクトレベルでレイアウトやライティングを扱うことに慣れたアーティストは、**Scene Import LOP** ノードを使用すると、ジオメトリ、ライト、カメラを簡単に LOP に取り込んでレンダリングできます。制御された USD シーングラフを作成したい場に合は、別のアプローチをお勧めしますが、Karma や他のレンダラに素早くアクセスするには Scene Import ノードが役立ちます。

USD 用にアセットを準備

Component Builder LOP ネットワークまたは **USD Export SOP** を使用して、プロップを構成する方法もあります。ジオメトリとマテリアルをプロップごとにセットアップしてから、レイアウト段階で使用できるように USD として書き出します。

プロップの一部には、**バリアント**を使用して、レイアウト時に選択するバリエーションを複数作成します。この USD の概念をセットアップできる LOP ノードがあります。

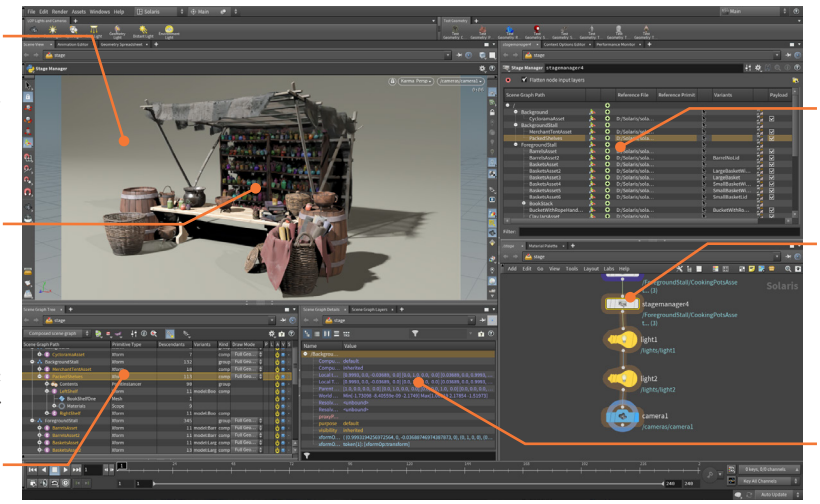
USD としてプロップをセットアップしたら、そのファイルを **Sublayer** や **Reference LOP** を使用してより大きいレイアウトシーンファイルに取り込み、**Edit LOP** を使ってプロップを配置します。アセットが適切に準備されていれば、ジオメトリ、マテリアル、バリアントがセットアップされ、準備が整った状態のアセットを使用できるようになります。

SOLARIS デスクトップ

ステージ - ステージビューは、レイアウトやライティングを確認したり、プリミティブ、ライト、カメラをインタラクティブに操作して適切にセットアップするための場所です。

ビューポートレンダー - ステージでは、Houdini GL または Storm を使用して、USD GL ソリューションをレンダリングできます。また、Karma や Hydra 準拠のサードパーティ製レンダラにインタラクティブにレンダリングすることも可能です。

シーングラフ - USD 構造は、レンダリング可能なシーングラフを提供し、このパネルを使用してグラフを検査できます。



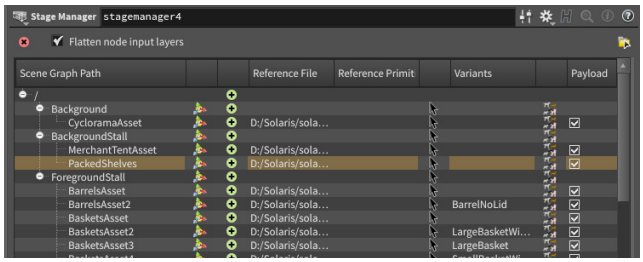
Stage Manager - このノードは、USD ファイルを読み込んで、ステージ上に配置することができます。

LOP ノード - LOP のすべてのアクションは、プロシージャルノードを使って実現されるため、簡単に戻って変更を加えられます。

Scene Graph Details - シーングラフでアイテムを選択すると、詳細を確認し、パイプラインでのステータスを把握できます。

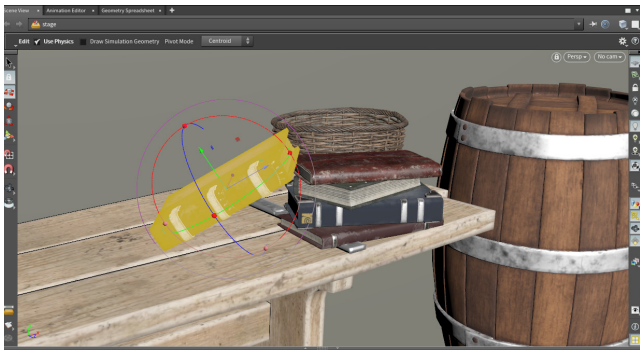
STAGE MANAGER

Stage Manager は、ディスク上の USD アセットを参照したり、それらを 3D 空間でトランスフォームしたり、シーン階層を調整するためのワンストップノードとして設計されています。これには、入力レイヤーを平坦化し、上流からの変更をブロックすることも含まれます。この柔軟性の欠如は、迅速なセットアップが可能という利点によってバランスされています。



物理ベースの編集

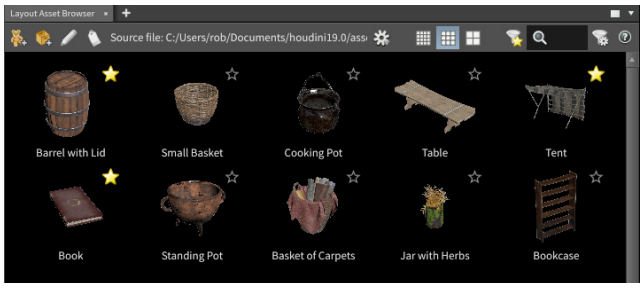
すべてのプロップをステージに読み込んだら、**Edit LOP** を使用してプロップを移動することができます。編集は別の非破壊レイヤーとなり、参照されたアセットは元のまま残るので、後で戻る必要がある場合も安心です。



Edit LOP の **Use Physics** オプションをオンにすると、Houdini のリジッドボディの機能を利用して衝突を検出したり、オブジェクトをリアルに配置できるようになります。3D ビューでインタラクティブに作業しながら、自然で有機的なルックを実現できます。

インスタンス化

USD には、LOP で利用できるインスタンス化ソリューションがあります。Instance LOP では、1 つまたは複数のオブジェクトを入力して、LOP 内でセットアップしたポイントに分散させることができます。これらのポイントは、シーンからジオメトリをインポートし、モデルから抽出したサーフェスにポイントをばら撒くことで作成されます。
マテリアルは、さまざまな方法でインスタンスに割り当てることができます。その方法の 1 つである **Material Variation LOP** では、ジオメトリごとにレンダリングプロパティを設定することも可能です。



また、**Layout Asset Browser** と **Layout LOP** を使用すると、ブラウザで選択したアセットを参照するインスタンスポイントをブラシワークフローで配置、編集、トランスフォームすることができます。

USD への書き出し

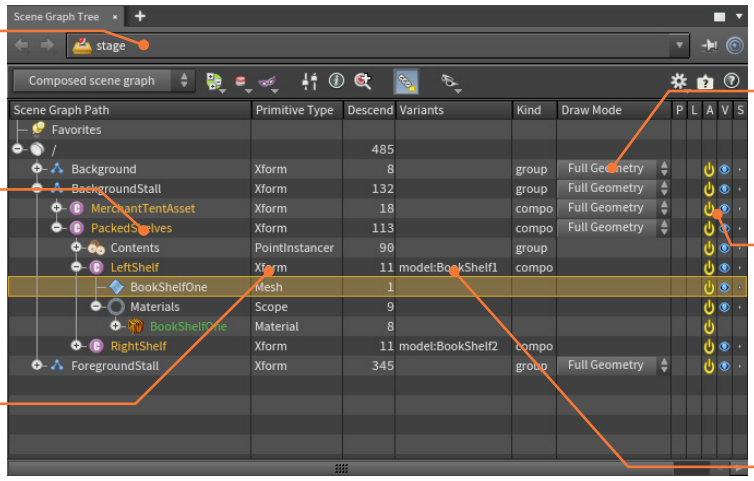
LOP グラフのさまざまなポイントで、ノードを RMB クリックして **LOP Actions > Inspect Active Layer** を選択することで、USD コードを検査できます。また、**平坦化したステージ**を検査することも可能です。USD に書き出すときは、すべてのレイヤーを分解することも、平坦化した単一のグラフとして保存することもできます。

シーングラフ

Stage - LOP を作成するためのトップレベルのネットワークコンテキストです。
また、この作業のすべてを LOP ネットワークで行えます。

Scene Graph Path - これらは、ステージのルックを定義する USD レイヤーおよびサブレイヤーを表します。ほとんどの場合、ディスク上のさまざまな USD ファイルがショットから参照されています。

Primitive Type - それぞれのプリミティブに、タイプ、つまり挙動を定義するスキーマがあります。これは、各レイヤーのステージに対する寄与を特定するのに役立ちます。



Draw Mode - パス内の任意の要素の表示を、Full Geometry、Bounding Box、Textured Cards のいずれかに変更できます。

ディスプレイオプション - 各レイヤーの可視性やアクティベーションを設定できます。シーングラフから何かを削除することはできないため、非表示にしたり非アクティブにする必要があります。

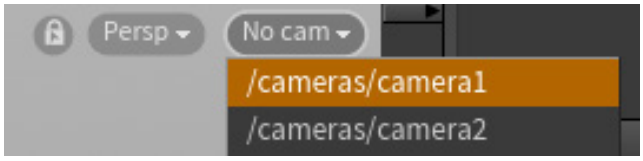
Variants - レイヤーにバリアントがある場合は、選択されているものがここに表示されます。

Solaris : カメラとライト

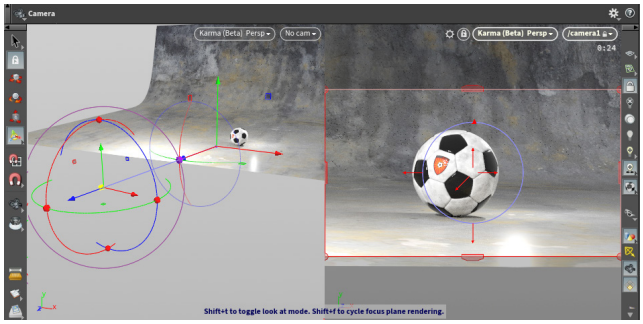
ショットをレンダリングする前に、カメラ越しにシーンを見て、シーンを照明する必要があります。Solaris/LOP コンテキストは、レイアウト、ルックデブ、ライティング、Karma によるレンダリング、オブジェクトレベルでは Mantra によるレンダリング向けに設計されており、ライトとカメラをセットアップできます。

カメラ

カメラは、**LOP Lights and Cameras** シェルフからアクセスができます。シェルフツールで **Alt クリック**すると、現在のビューがカメラビューに変わります。ネットワークビューで Camera ノードを作成した場合は、ビューポートの左上にある **No cam** メニューをクリックすると、カメラ越しに見られるようになります。



カメラを調整するには、別のビューから、またはカメラ越しに見ながら、カメラハンドルを使用して操作します。Display Options バーには **lock camera to view** ボタンがあり、**View ツール**を使ってカメラの位置を変更できます。



このオプションはオンのままにしないでください。オンのままにすると、ビューを変更しただけでカメラビューが変わってしまうことになります。気に入ったカメラビューは、**トランスフォーム値をロック**して、ビューを変更しても失われないようにすることをお勧めします。

カメラプロパティ

LOP コンテキストのカメラには、カメラの画像生成方法を決める主なプロパティがあります。

View タブ

- **Projection** - 遠近法または正投影のどちらにするかを選択します。
- **Focal Length** - 焦点距離を選びます。値が小さいほどワイドなショット（広角）に、値が大きいほどロングショット（望遠）になります。
- **Horizontal/Vertical Aperture** - 絞りは、カメラに入る光の量を制御するゲートです。

Sampling タブ

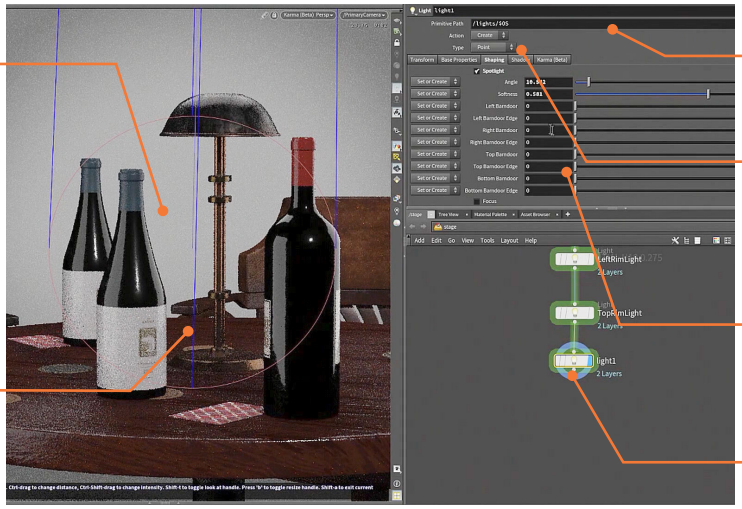
- **Shutter Open/Close** - シャッターを開いておく時間を決定します。この設定は、モーションブラーに影響します。
- **Focus Distance** - カメラから焦点平面までの距離。Depth of Field を使用している場合には、どのオブジェクトを鮮明に見せる（焦点にする）かを決定します。
- **F-Stop** - レンズの絞り。デフォルトは 0 で、焦点の範囲を無効にしています。

Shift + F を押すと、焦点平面が表示され、どのようにジオメトリと交差しているかが分かります。カメラ越しに見ているときに、サーフェスを **Shift クリック**またはドラッグすると、そのポイントと交差するように焦点平面を動かせます。カメラ視点外のときは、焦点平面上のハンドルを使用してその平面を動かし、被写界深度を設定します。

ライティングセットアップ

ビューポートレンダリング - ライティングを決定するには、Karma または RenderMan や Arnold などのレンダラを使用してビューポートでレンダリングできることが重要です。HoudiniGL も使用できますが、これらのメインレンダラほど効果的ではありません。

ライトハンドル - カメラと同じように、一歩下がってビュー内のライトを操作します。専用のコントロールを使用して、カメラビューでライトを設定することも可能です。



Primitive Path - USD シーングラフ内のライトの場所を設定します。

ライトタイプ - このメニューでは、ライトタイプを選択します。Light LOP では、すべてのライトタイプにアクセスでき、切り替えもできます。

ライトパラメータ - 円錐角や強度など、ライトのプロパティを制御するパラメータが幅広く用意されています。

Light LOP ノード - ライトは個々に、LOP ノードとしてネットワークに追加されます。

ライト

ライトにはシェルフからもアクセスでき、同様のハンドルを使って配置できます。Houdini には、さまざまなライトタイプが用意されています。

Point Light - ポイントから全方向にライトを放出します。電球にも似た、いわゆる点光源です。

Spot Light - 円錐形のライトのビームを、ポイントから特定の方向に放射します。

Area Light - 指定した領域上に、たくさんの光源を自動的に分散します。Line、Tube、Grid、Disk、Sphere の 5 つの形状のエリアライトから選択できます。

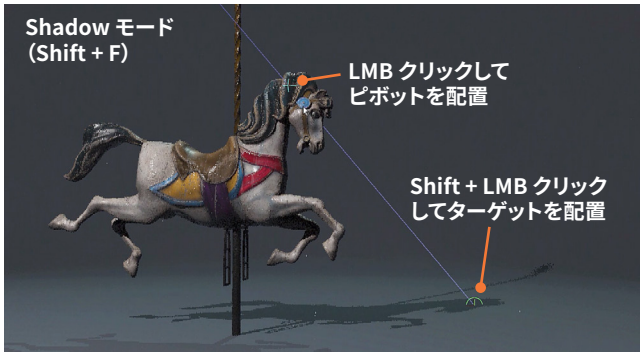
Geometry Light - ジオメトリオブジェクトのサーフェスシェーダを使用して色を付けたライトを、シーンに放出します。

Distant Light - 平行光線を放射します。このライトは、太陽光線に似ています。

Environment Light - 半球または球の環境からのライトをシーンに投影します。

カメラビューでのライティング

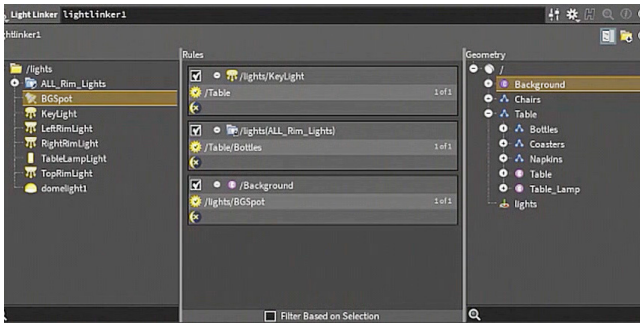
Light または **Light Mixer LOP** を選択および表示している場合、カメラ越しに見ながら、ライトのプロパティの多くを設定することができます。**Specular (Shift + S)**、**Diffuse (Shift + D)**、**Shadow (Shift + F)** オプションを使用すると、シーン内のサーフェスをクリックしてライトをセットアップできます。



その後、**Ctrl ドラッグ**して、ショットからライトの**距離**を変更したり、**Ctrl + Shift ドラッグ**して**明るさ**を変更することができます。ビューポートでこうした操作を行うと、ハンドルをドラッグするために離れる必要がないため、作業中のショットに集中できます。

LIGHT LINKER

ライトを特定のオブジェクトにリンクするのは、ショットのライティングを制御するのに最適な方法です。Solaris では、**Light Linker LOP** を使用してリンクできます。このノードには、オブジェクトとライトを接続するためのインターフェースが含まれています。



ライトコレクションを使用すると、プリミティブとライト間の相互作用を定義したルールを用いて、より効率的にリンクを適用できます。

ライトのインスタンス化

Solaris/LOP コンテキストでは、Houdini のプロシージャルなジオメトリノードを使用してポイントを作成し、それらのポイントに**ライトをインスタンス化**できます。その後、ポイントにアトリビュートを追加して、例えば、輝度を順に変化させて、レトロなマーキー看板のようなエフェクトを作成することができます。このアプローチなら、ライトをセットアップしやすいうえ、エフェクトを追加したり、ショットのニーズに合わせて変更することも非常に簡単です。

LIGHT MIXER

ライトリスト - このリストには、ミキサーに接続しているすべてのライトが表示されます。

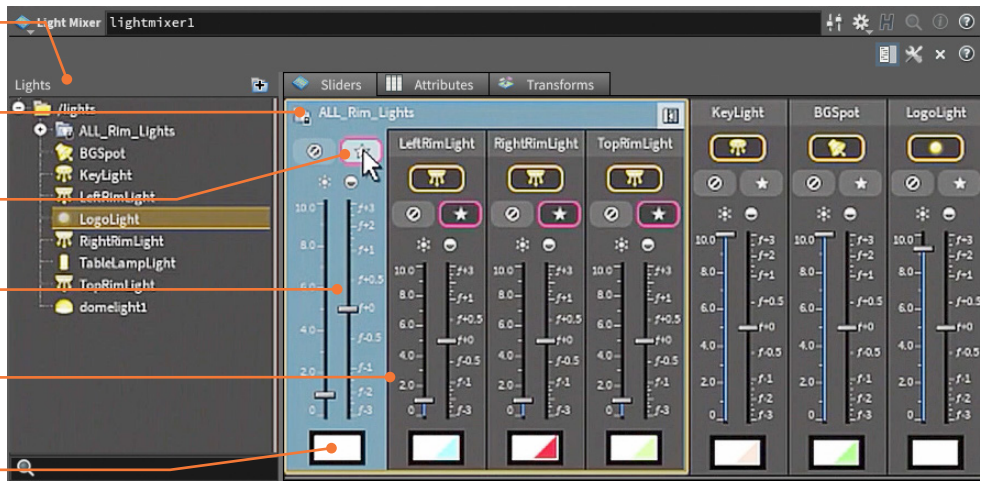
コレクション - ライトをコレクションにまとめると、ミキサー内でグループとして機能させることができます。

ソロ - 星アイコンをクリックすると、ライトまたはコレクションをソロにすることができます。

強度スライダ - 1 つ目のスライダで、ライトまたはライトコレクションの強度を制御できます。

露出スライダ - 2 つ目のスライダは、露出を制御します。強度の微調整に利用できます。

ライトカラー - ここをクリックして、ライトまたはライトコレクションに色付けします。



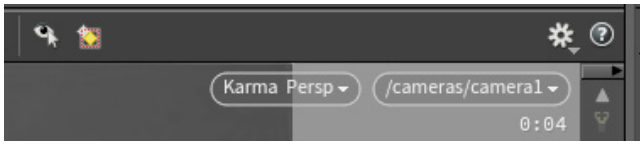
レンダリング

ショットをレンダリングするとは、カメラとライトを使用して 3D オブジェクトをデジタルで撮影し、画像または画像シーケンスを生成するようなものです。ゲームアーティストは、ゲームのシネマティクスのためにレンダリングすることもあれば、高解像度から低解像度のオブジェクトにテクスチャをベイクするために、レンダリングすることもあります。

KARMA

Karma は物理ベースのレイトレーサーで、Solaris/LOP コンテキストで USD ファイルを扱えるよう設計されています。CPU 上で動作し、ディスプレイスメントおよびサビディビジョンサーフェスのためのアダプティブ（適応型）テッセレーション、マルチセグメントのモーションブラー、インスタンス化、ヘアーとファー、強力なボリュームレンダリングのサポートといった機能を備えています。

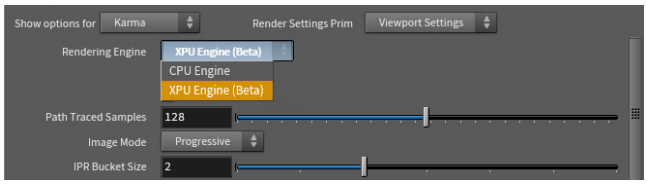
Karma は、USD イメージングフレームワークである **Hydra** で機能します。ビューポートで使用してインタラクティブに更新したり、Karma ノードを使用してディスクにレンダリングすることができます。



Karma は、VEX、USD Preview、Material X で作成したシェーダを扱えます。

Karma XPU

Houdini 19.5 には、**Karma XPU** レンダリングエンジンのベータ版が含まれています。この GPU/CPU のハイブリッドレンダラは、アルファ版としてリリースされています。多くの機能は開発中であるため、このエンジンはテスト目的でのみ使用します。XPU は、Scene View の Display Options、または Karma ノードで選択できます。



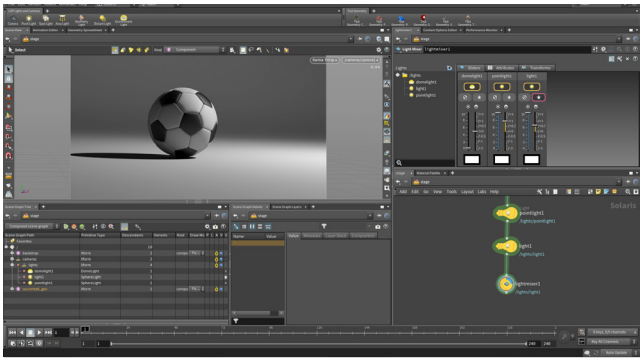
Karma XPU では、USD Preview シェーダと Material X を扱えますが、VEX には対応していません。

サードパーティ製レンダラ

USD をサポートする Solaris では、**RenderMan**、**Autodesk Arnold**、**V-Ray**、**Maxon RedShift**、**AMD ProRender** といった他の Hydra デリゲートにレンダリングすることが可能です。

ビューポートレンダリング

Karma レンダラの主な利点の 1 つは、パースビューで使用できることです。Karma を選択することで、インタラクティブなレンダリングが可能となり、Solaris/LOP コンテキストでライティングやルックデブに関する決定を下すことができます。

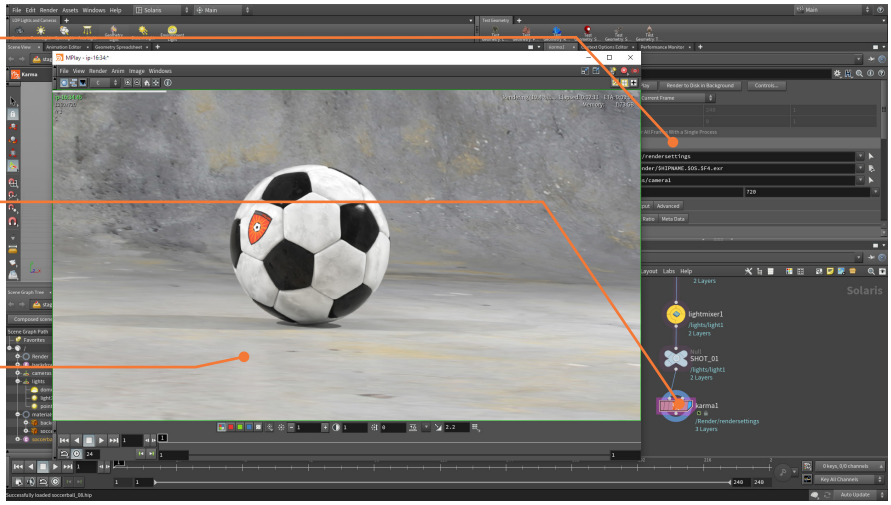


ビューポート | KARMA

レンダリング設定 - Karma LOP を使用してレンダリング設定を定義し、ディスクにレンダリングできます。ここでは、ディスクにレンダリングする際のパスや、カメラ設定などを設定します。

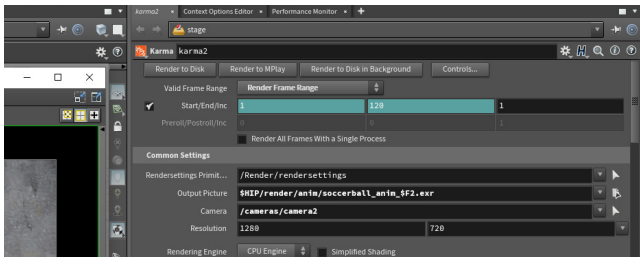
Karma LOP - Karma LOP を Solaris ノードネットワークの終端に追加します。この LOP のさまざまなバージョンを使用することで、テストレンダリングや最終ショットなど、それぞれ異なる結果をセットアップできます。

MPlay - Karma と Mantra の両方から MPlay に直接レンダリングできます。または、ディスクにレンダリングしてから MPlay で開き、結果を確認することも可能です。



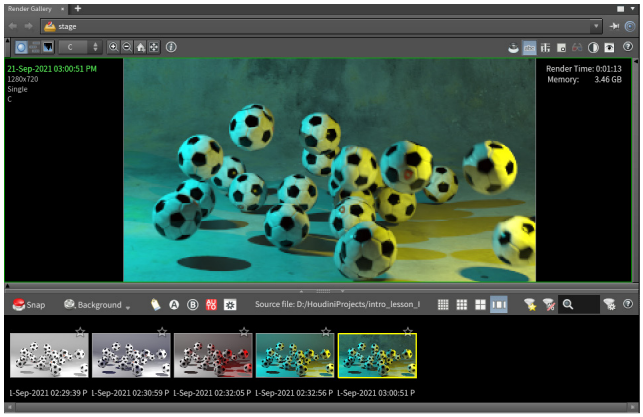
レンダリング設定

ステージをレンダリングするときは、ビューポートのレンダリング設定を使用します。レンダリングの最終的なルックを作成するには、**Karma LOP** を使用してフレーム範囲、カメラ解像度、デノイザ、より高画質のレンダリング設定を設定します。



RENDER GALLERY

Render Gallery を使用すると、**スナップショット**を撮って進行状況を確認できます。各スナップショットには、ルックのすべての設定が含まれ、そのスナップショットと合致するようにシーンをいつでも戻すことができます。スナップショットにはラベルが付けられ、フィルタリングできるので、アクセスも簡単です。

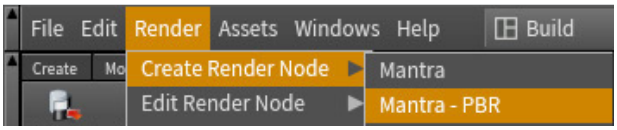


MANTRA

Mantra は、Solaris が導入される前に開発された Houdini のレンダラです。ジオメトリ、インスタンス、ボリュームの高効率なレンダリングが密に統合された、物理ベースのレンダリングエンジンですが、Solaris/LOP では動作しません。

出力ノード

ショットをレンダリングするには、レンダリング出力ノードを作成する必要があります。このためには、**Render > Create Render Node > Mantra - PBR** を選択します。



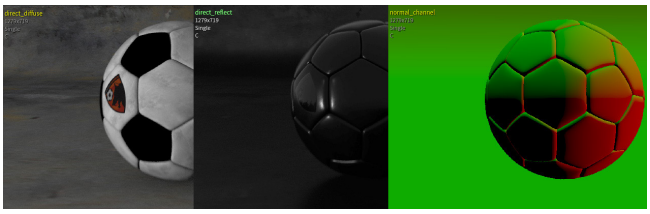
また、**Tab** キーを使用して出力ネットワークに **Karma ROP** を追加することもできます。この中には、オブジェクトレベルからすべての可視オブジェクトを取得する LOP ネットワークが含まれています。

ROP ノードを使用すると、ディスクまたは Mplay にレンダリングすることができます。これらのノードには、サンプリング、ノイズレベル、全体的なレンダリング品質など、最終画像を制御するためのパラメータが多数含まれています。

オブジェクトやオブジェクトのグループごとに異なる ROP を設定できます。異なるノードを接続することで、ROP の依存関係を作成できます。チェーンの最後のノードでレンダリングボタンを押すと、残りのすべてのノードが先にレンダリングされます。

レンダリング出力 | AOV

ROP には、**画像平面**をセットアップするためのコントロールが用意されており、***Direct Lighting***、***Indirect Lighting***、***Shadows***、***Depth*** などのレンダリングレイヤーを作成できます。Karma と Mantra の両方がこれらのパスをレンダリングでき、Houdini のコンポジットコンテキストである COP や、Nuke などの外部コンポジットツールで読み込めます。



Background Plate LOP を使用すると、背景が見透けるようにシーンに穴をつくるマットオブジェクトをセットアップできます。このジオメトリは、影を受けたり、光に反射するため、オブジェクトがリアルに馴染みます。

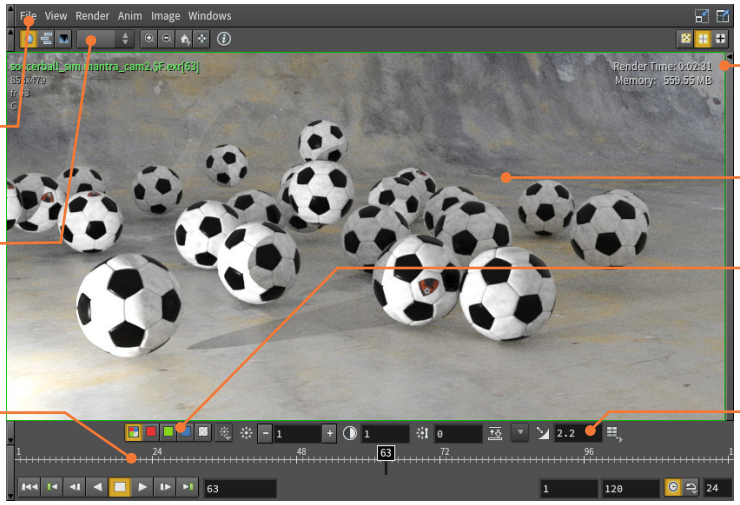
MPLAY

MPlay では、Karma や Mantra などのレンダラでレンダリングした画像を確認できます。

メインメニュー - 画像または画像シーケンスをプレビュー用に読み込みます。別フォーマットで保存することも可能です。

レンダリングレイヤー - このメニューを使用すると、***color***、***normal***、***diffuse_direct***、***reflect-direct*** など、さまざまなレンダリングレイヤーを表示することができます。

プレイバー - 画像シーケンスを読み込んだ場合、これらのコントロールを使用して、シーケンスを再生したり、スクラブすることができます。



Render Time - このビューに直接レンダリングする場合もあるため、レンダリング時間情報が表示されます。

表示オプション - **MMB ドラッグ**でパンしたり、**RMB ドラッグ**でズームイン/アウトできます。

チャンネル - これらのボタンをクリックすると、赤、緑、青、またはアルファチャンネルに集中したり、それらの組み合わせを確認することができます。

ガンマ設定 - ビューポートの輝度、コントラスト、ガンマを設定できます。デフォルトでは、リニアワークフローをサポートするガンマ値 2.2 が使用されます。

時間とモーション

アニメーションは、経時的な変化を伴います。オブジェクトの位置、形状、色など、何であれ、時間の経過とともに変化させれば、それはアニメーションです。Houdini には、キーフレームベースのワークフロー向けのさまざまなツールに加えて、時間とモーションをより高度に操作できる Motion FX と CHOP が用意されています。

キーフレームの設定

キーフレームを使用すると、特定の時点に特定のパラメータ値を設定することができます。パラメータ値が変化すると、シーン内のオブジェクトがアニメートされます。その後、アニメーションカーブを使用して、キーフレーム間のモーションの品質を決定できます。以下に、Scene View で作業しながらオブジェクトにキーフレームを設定できる主なホットキーを紹介します。

- **キーフレームの設定** K
- **AutoKey の切り替え** Alt + K
- **ハンドルをキーフレーム** Ctrl + K
- **位置をキーフレーム** Shift + T
- **回転をキーフレーム** Shift +R
- **スケールをキーフレーム** Shift + E

パラメータエディタでキーフレームを設定することも可能です。**Alt** キーを押しながらパラメータ名またはパラメータフィールドをクリックするか、パラメータを **RMB クリック**して **Keyframes > Set Keyframe** を選択します。こうすると、一度に 1 つのパラメータにキーフレームを設定できます。

プレイバー

プレイバーはメインのワークスペースの下部にあり、アニメーションを再生したりスクラブできます。時間はフレーム単位で、デフォルトのフレームレートは 24 フレーム/秒です。
左側には再生コントロールがあります。素早く再生をセットアップしたり、時間を操作するホットキーをいくつか紹介します。

- **再生** ↑
- **逆再生** ↓
- **次のフレーム** →
- **前のフレーム** ←
- **開始フレーム** Ctrl + ↑
- **次のキーフレーム** Ctrl + →
- **前のキーフレーム** Ctrl + ←

プレイバーで、キーフレームを編集することもできます。プレイバーでフレーム範囲を **RMB クリック**すると、キーを **Cut**、**Copy**、**Paste** するオプションに加え、**Replace**、**Cycle**、**Repeat**、**Stretch** といった特殊なペーストにもアクセスできます。これらのオプションにも独自のホットキーがあり、メニューに記載されています。**アニメーションエディタ**に移る前に、プレイバーでさまざまな作業を行うことが可能です。

チャンネル

アニメーションエディタでキーフレームを設定したり、アニメーションカーブを表示するときには、**チャンネル**を操作することになります。キーフレームを設定したチャンネルを持つオブジェクトを選択すると、そのチャンネルがアクティブになり、キーフレームが**プレイバー**や**アニメーションエディタ**に読み込まれます。オブジェクトを選択解除すると、チャンネルをピン留めしない限り、それらのチャンネルも選択解除されます。
チャンネルをピン留めするには、**プレイバー**の右側、**アニメーションエディタ**の左側、または**チャンネルリスト**ペインにある**チャンネルリスト**を使用します。このリストで 1 つまたは複数のチャンネルを選択することで、キーフレームを設定したり編集する対象のチャンネルを絞り込みます。

チャンネルリストペイン

チャンネルリストペインでは、**チャンネルグループ、アニメーションレイヤー、アクティブなチャンネル**を扱うことができます。リストを使用してチャンネルのグループを作成すれば、アクセスがしやすくなります。また、グループを使ってチャンネルをピン留めすると、異なるオブジェクトを選択した場合でもそのチャンネルを操作できます。これは、キャラクタにキーフレームを設定する場合に便利なペインです。

Flipbook

シーンをアニメートするとき、そのモーションをプレビューしたいことがあります。Scene View の左側のツールバーにある **Flipbook** ツールを使用すると、ビューポートからフレームをキャプチャして、その結果をリアルタイムでムービーとして再生できます。

プレイバー

プレイバーを使用すると、時間をスクラブしたり、キーフレームを設定および編集することができます。
プレイバーでも簡易の編集が可能ですが、細部まで調整する目的ではアニメーションエディタを使用します。

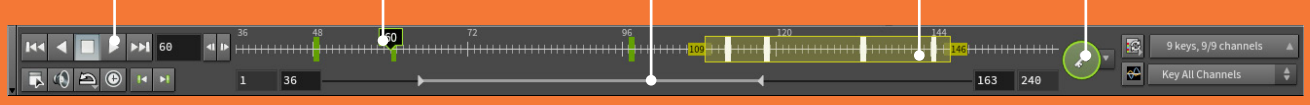
再生コントロール
再生、一時停止、次のキーへの移動を素早く行えます。その下には、アニメーションオプションやリアルタイム再生のボタンがあります。

現行時間
現行時間は、フィールドとフレーム範囲の黒のマーカー上に表示されます。マーカーを使用してプレイバーをスクラブできます。

フレーム範囲
全体の範囲は、左端のアニメーションコントロールボタンで定義します。表示範囲は、範囲の下部にあるハンドルを使って縮小することができます。

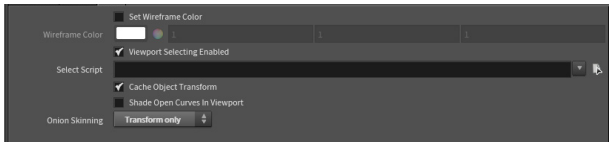
キーの編集
キーを設定すると、そのキーがプレイバーに表示されます。**Shift** キーを押しながら **LMB** でキーを選択してから、**MMB** でドラッグしてキーを編集します。

キーの設定
Set Key ボタンをクリックして、キーフレームを設定します。小さい矢印をクリックすると、**Auto キー**などのオプションメニューが表示されます。



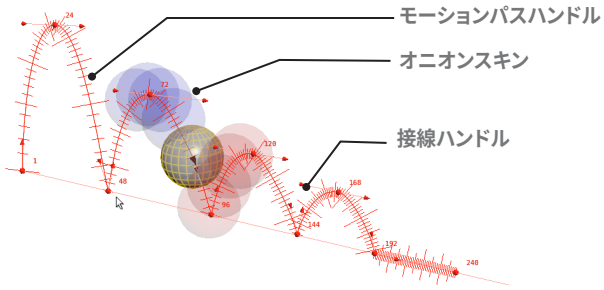
ONION SKINNING

Onion Skinning を使うと、現行のフレーム前後のフレームのオブジェクトをゴーストとして表示できます。Onion Skinning は、オブジェクトの **Misc** タブでオンにします。Frames Before、Frame After、Frame Increment などのオニオンスキンのオプションは、ビューポートの **Scene** タブの Display パネル (**d**) で確認することができます。



モーションパスハンドル

Pose ツールを使用してアニメートする場合、**Motion Path** オプションをクリックしてハンドルを表示すると、選択したオブジェクトのアニメーションを時間経過に応じて確認できます。また、ハンドルを使用してモーションの形状を操作することもできます。



アニメーションエディタ

選択したチャンネルはアニメーションエディタに読み込まれ、キーフレームとアニメーションカーブ、またはスプレッドシートやドープシートとして表されます。キーフレームは選択と編集が可能で、カーブの形状は接線ハンドルを使って調整できます。カーブはキーフレーム間のモーションを定義するもので、モーションの質を定義するうえで重要な役割を果たします。

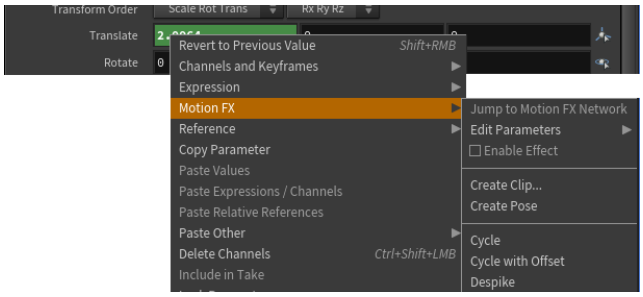
チャンネルを操作するときには、以下のホットキーを使ってキーフレームやアニメーションカーブを確認できます。

- **すべてを表示/ホーム** H
- **パン** MMB
- **ズーム** RMB

Motion FX

キーフレームとアニメーションカーブはノードのパラメータに格納されますが、**チャンネルオペレータ (CHOP)** を使用すると、よりプロシージャルなノードベースのアプローチでモーションを操作することができます。

最も簡単にチャンネルオペレータを作成するには、パラメータを **RMB クリック**して、**Motion FX** サブメニューから選択します。チャンネルリストを使用して、これらのエフェクトをチャンネルグループに適用することも可能です。



Motion FX は、**Channel CHOP** に抽出および格納される、キーフレームによる動きに適用できます。その後、**Cycle**、**Noise**、**Smooth**、**Limit**、**Lag** などのエフェクトを既存の動きに適用します。**Constraints** シェルフにはさまざまなツールがあり、パラメータ設定によってターゲットの方を**向くようにしたり**、**遅延させたり**、**微震する**ようにできます。

こうしたノンリニアなアプローチなら、非常に柔軟かつユニークな方法でモーションを扱えます。

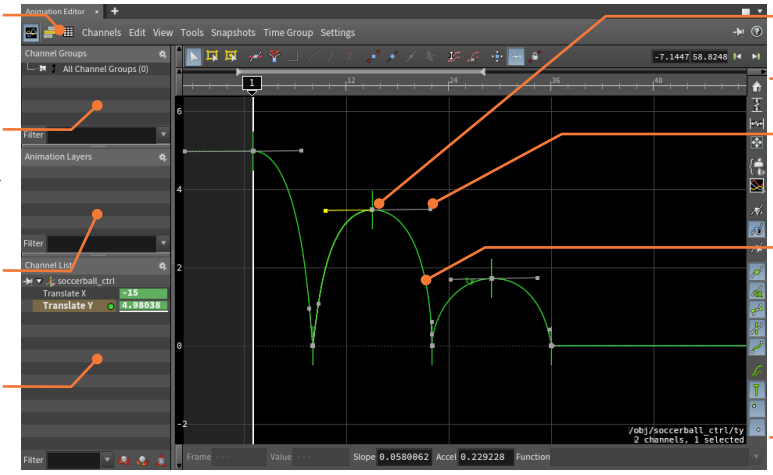
アニメーションエディタ

エディタオプション - このエディタは、グラフ、ドープシート、テーブルの表示を切り替えられます。

Channel Groups - グラフのこの領域にはチャンネルグループが表示され、簡単にチャンネルを選択したりピン留めできます。

Animation Layers - この領域では、複数チャンネルを重ねて、異なるイテレーションを作成できます。

Channel List - ここには選択したオブジェクトのチャンネルが表示されます。グラフで確認したいチャンネル名を選択します。



キーハンドル - 垂直バーを使用してキーを前後の時間に移動したり、ボックスを使用して値を編集できます。

接線ハンドル - カーブの形状を調整するために、キーフレーム前後の接線の角度を定義します。

カーブ - アニメーションカーブは、キーフレーム間のモーションを定めます。これによってモーションの質が決まります。

カーブ関数 - これらを使用して、アニメーションカーブやハンドルのディスプレイオプションを設定できます。

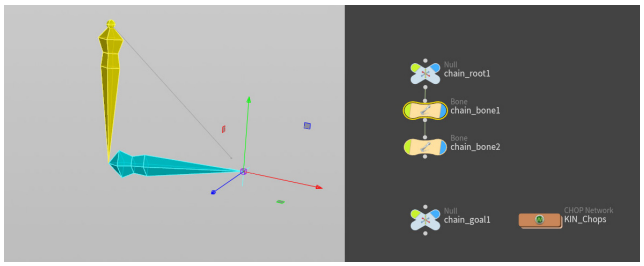
キャラクタリギングと FX

Houdini には、キャラクタやクリーチャを作成するための広範なリギングツールが搭載されており、それらを Houdini デジタルアセットにラップしてアニメータに渡すことができます。さらには、ヘアー、ファー、マッスル、布、群衆など、キャラクタのルックを向上させるためのキャラクタ FX ツールも備わっています。

BONES

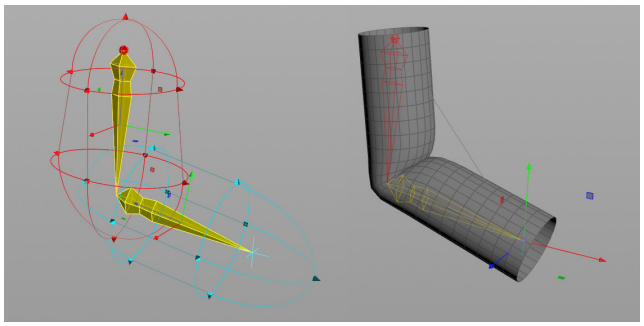
Houdini では、**Rigging** シェルフにある **Bones** ツールと **Bones from Curve** ツールを使用して、ボーンを描画および編集できます。ボーンチェーンはそれぞれ、チェーンルートとボーンで構成されています。他の 3D アプリがジョイントベースであるのに対し、Houdini は **Length** や **Rest Angle** のパラメータを持つ **Bone** ノードを使用します。

また、**Bones** ツールを使用すると、チェーンにインバースキネマティクスを追加して、**エンドエフェクタ**や、場合によっては**ツイストエフェクタ**を加えることも可能です。キネマティクスは、独自のサブネットワークに存在する**チャンネルオペレータ**、つまり **CHOP** ノードによって駆動されます。



ジオメトリのキャプチャ

キャラクタのジオメトリをボーンに**キャプチャ**して、リアルな動きを表現するために必要な変形を作成できます。Houdini のボーンには **Capture Region** があり、ジオメトリを包含するようセットアップできるうえに、ジョイント部分には適切な重なりが作られます。このプロセスにより、ポイントにウェイトアトリビュートが割り当てられ、ボーンが移動したり回転したりするときにジオメトリを制御する **Deform** ノードに供給されます。



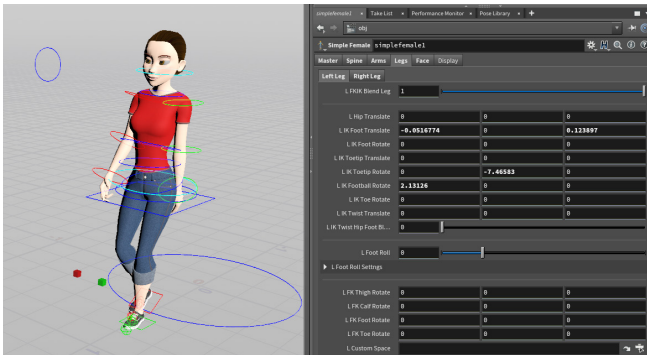
最初にキャプチャしたジオメトリは、希望するルックとは異なるでしょう。さまざまなツールを使用して、**キャプチャウェイトを編集したりペイントする**ことで調整していきます。ジョイントのウェイトを滑らかにして、曲げをリアルにしていきます。また、**Deform** ノードに接続した **DeltaMush** ノードを使用して、ポイント変形の効果を平滑化することも可能です。

Bone Capture Biharmonic という新しいテクニックを使用すると、広範なポイントウェイトがなくても、ジオメトリをキャプチャして、ジョイント部分を希望通りのルックにすることができます。この方法では、四面体メッシュに Biharmonic（重調和）関数をセットアップすることで、より全体的なソリューションが作成されます。

デジタルアセットキャラクタ

Houdini キャラクタをリギングしてアニメーションチームと共有するには、ボーン、ジオメトリ、マテリアルを **Houdini デジタルアセット**にラップする必要があります。

これにより、ファイルがディスク上に作成され、アニメータは複数のショットで簡単に参照できます。ハンドルやキーパラメータはトップレベルで操作が可能で、アニメータはリグの内部構造を気にすることなく、キーフレームを設定できます。また、Pose Library や Character Picker のセットアップをアセットに保存して、素早くアクセスできるようにすることも可能です。

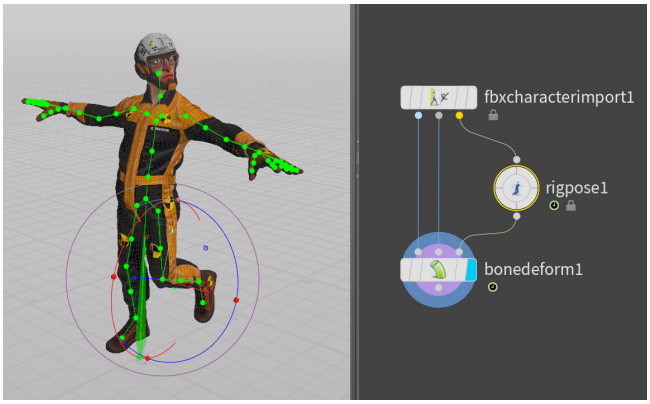


キャラクタのパーツに加えた変更はアセットに保存され、すべてのショットが更新されます。この仕組みにより、堅牢かつ管理しやすいキャラクタパイプラインが実現します。

KINEFX

KineFX は、リターゲットやモーション編集にプロシージャルな基盤を提供するキャラクタツールセットです。将来は、リギングやアニメーションにも拡張の予定です。ジオメトリコンテキストにおけるこの新しいワークフローは、限りない柔軟性とキャッシュ機能を備え、迅速かつプラグアンドプレイのリギング体験を実現します。

ジオメトリ（SOP）コンテキストに実装された KineFX は、ジョイントを通常のポイントジオメトリとして扱い、エッジ接続によってリグの階層を定義します。Houdini のオブジェクトレベルからリグを取り込むことも、FBX キャラクタを読み込むことも可能です。



チャンネルグループ

Houdini でアニメートする際は、スコープされているチャンネルにキーフレームを設定したり、**アニメーションエディタ**に表示することができます。通常は、現在選択しているチャンネルがこれに該当します。また、**チャンネルグループ**はまとめることも可能です。チャンネルをスコープしたりピン留めすると、キーフレームを設定しやすくなります。

キャラクタをデジタルアセットとしてセットアップしてある場合は、パラメータエディタの左上にあるアイコンをクリックし、**Parameters and Channels > Create Nested Channel Groups** を選択して、アセットのフォルダ階層をガイドとしてグループを作成します。適切に設計されたキャラクタアセットであれば、この作業は簡単です。

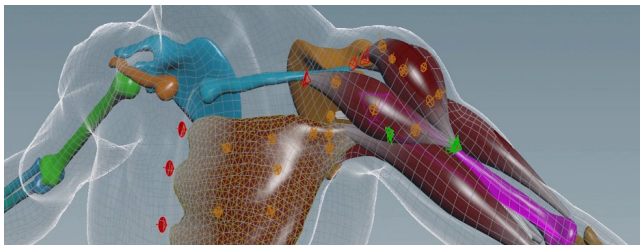
キャラクタ FX | ヘアーとファー

Add Hair ツールでキャラクタにヘアーを追加したら、Houdini のヘアーとファーのツールセットを使ってセットアップとグルーミングを行います。これらのツールセットではガイドヘアーも使用できます。ワイヤーシミュレーションでアニメートすると、リアルな仕上げになります。

キャラクタ FX | マッスルとスキン

Houdini では、アニメートしたクリーチャにマッスルを追加し、シミュレーションを実行することなく、それらをスキンデフォーマとして適用できます。まず、ジオメトリコンテキストで **Muscle** ノードを使用して、シンプルなマッスルのフォームを作成します。

その後、マッスルの形状と位置を調整し、キャラクタリグに取り付けたら、自動セカンダリアニメーションまたは Jiggle（微震）を有効にします。Houdini のマッスルシステムは、統合されたデジタルアセットを使用しながら、FEM（ダイナミクスシミュレーション）および非 FEM（スキンデフォーマ）ワークフローに対応できるように設計されています。



群衆シミュレーション

群集シミュレーションは、キャラクタのスケルトン、スキンジオメトリ、アニメーションクリップで構成されるエージェントから始まります。これらはポイントに割り当てられ、シンプルなルールの組み合わせによって複雑な挙動を作り出します。また、エージェントは他の動的要素と相互作用が可能です。例えば、エージェントが走り過ぎる車にぶつかってラグドールになったり、群衆がフィールド上のアクションに反応するようトリガを設定したりできます。

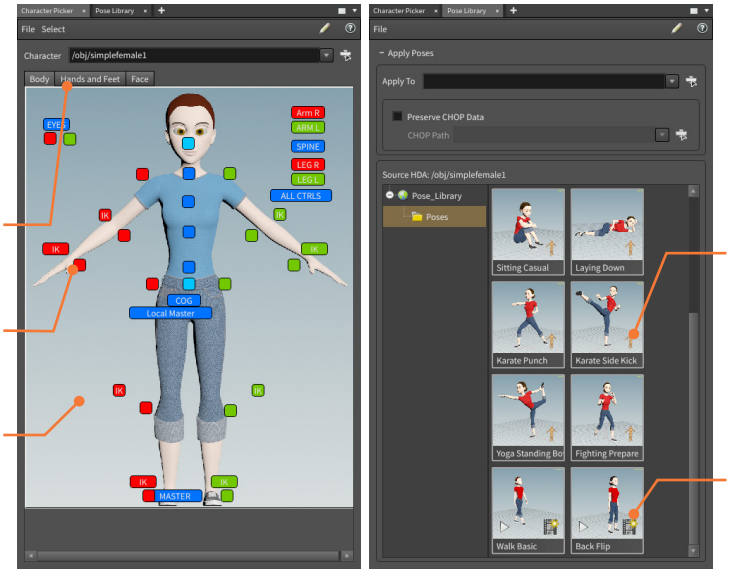
CHARACTER PICKER

このペインでは、リグのパーツを選択するためのインターフェースを作成できます。これをファイルに保存して、ディスク上のデジタルアセットファイルに追加することも可能です。

タブ - 手、足、顔など、体のパーツごとにタブをセットアップできます。

コントロール - リグの各ハンドルにマーカーを配置し、テキストやカラーで区別できるようにします。

背景画像 - キャラクタの視覚表現を追加して、マーカーと体のパーツを適切に関連付けられるようにします。



POSE LIBRARY

Pose Library ペインでは、後で参照できるように、キャラクタのポーズやクリップをキャプチャすることができます。ポーズを適用するには、プレイバーでフレームに移動し、ここで目的のポーズをクリックします。

ポーズ - 単一フレームから取り込んだポーズが保存されています。そのポーズに設定されているすべてのパラメータが、現在のシーンのキャラクタに適用されます。このポーズから別のポーズへの遷移を補間することも可能です。

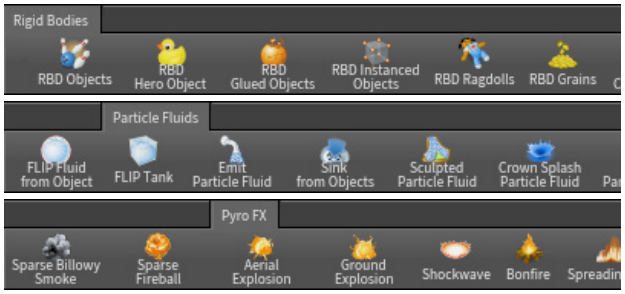
クリップ - クリップには、一定期間にわたるキーフレームが保持されています。例えば、歩行サイクルやバク転などの特徴的な動きが含まれます。

ダイナミックシミュレーション

Bullet のリジッドボディ破壊、Pyro FX の火と煙、Vellum ソフトボディ、FLIP 流体などを作成するとき、Houdini では、統合されたダイナミクス環境を利用できます。異なるソルバ間で相互にやり取りできるため、結果についての制御がしやすい環境です。

シェルフツール

ダイナミックシミュレーションのセットアップには、**ダイナミクスまたは DOP** コンテキストのノードと、**ジオメトリまたは SOP** コンテキストのノードから成るネットワークが必要です。シェルフツールを使用すると、これらのノードをすべて追加できるうえ、より少ないクリック数でシミュレーションをセットアップできるので、とても便利です。セットアップしたら、ネットワークの中に入り、それぞれのノードを確認できます。



シェルフツールは、ノードグループを自動的にセットアップしたい場合に適しています。シェルフツールによって構築されたネットワークの構成を確認しておくと、DOP ネットワークをゼロからセットアップする際に役立ちます。

ダイナミクスソルバ

シミュレーションの中心は、ダイナミクスソルバです。シミュレーションの頭脳として、すべてのダイナミクスオブジェクト、フォース、衝突オブジェクトを受け取り、それらを統合して最終結果を作ります。シェルフツールは、ソルバをダイナミックネットワークに組み込み、ノードを接続してくれます。

Rigid Body Solver - 効率的な Bullet ソルバや Houdini の組み込みソルバを使用して、リジッドオブジェクトの落下や衝突をシミュレートします。

Static Solver - オブジェクトを衝突ジオメトリとして機能させたいが、シミュレーションの影響を受けないようにしたい場合に使用します。

Flip Solver - FLIP 流体シミュレーションによって、しぶきや波のエフェクトを作成します。

Whitewater Solver - FLIP の計算が完了した後、このソルバを実行することで Foam（泡沫）、Spray（飛沫）、Bubble（泡）を作れます。

Vellum Solver - POP Solver の一種で、布、ヘアー、粒、流体、風船などのソフトボディに対して統合的に対応しています。

POP Solver - パーティクルや粒に使用され、さまざまなパーティクルベースのシナリオを幅広くシミュレートします。粒のシミュレーションは、ソフトボディや布の類のシミュレーションにも使用できます。

Wire Solver - ヘアーやファー、船の索具や木の枝などのワイヤー状のオブジェクトに使用できるソルバです。

Finite Element Solver - 連続体や四面体で定義されるソリッドの力学をシミュレートします。このソルバは、筋肉、ソフトボディシミュレーション、折れる木をはじめとする破壊ショットに使用されます。

Cloth Solver - キャラクタなどの変形ジオメトリと衝突する布シミュレーションを作成します。

SOP Solver - SOP ネットワークを使用して、オブジェクトの形状を時間の経過とともに変化させます。オブジェクトがぶつかってへこむ壁などをシミュレートできます。

OPENCL

POP Grains ノード、**Pyro ソルバ**（**Advanced** タブ）、**FLIP Solver**（**Volume Motion > Solver** タブ）などのソルバで **OpenCL** を使用する場合、GPU を使用するとシミュレーションが高速化できます。

フォース

ダイナミックな動きを作るには、動きのためのフォースが必要です。最も基本的なフォースは重力ですが、扇風機、流体、磁石などの外部フォースも、シミュレーションの動きに寄与します。

Gravity Force - オブジェクトが重力フィールド内にあるかのように、オブジェクトに下向きのフォースを適用します。

Drag Force - 現行のモーションに抵抗するフォースと回転モーメントをオブジェクトに加えることで、減速させたり、運動量を小さくしたりします。

Uniform Force - 正確な量のフォースと回転モーメントをオブジェクトに適用します。Noise DOP によって増強し、乱流を追加することもできます。

Fan Force - 円錐状のフォースをオブジェクトに加えます。

Fluid Force - 流体によって布やワイヤーなどのソフトボディを変形します。

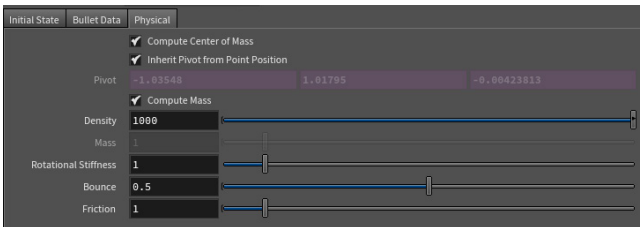
Wind Force - 押す力によってオブジェクトの Velocity を上げますが、風自体の速度を越えることはありません。

Magnet Force - メタボールで定義したフォースフィールドを使って、オブジェクトを引き寄せまたは遠ざけます。

Vortex Force - 渦巻きのようなフォースを作成して、竜巻周辺のオブジェクトのように、カーブを軸にオブジェクトを周回させます。

ダイナミクスオブジェクト

オブジェクトを選択し、シェルフツールを使ってシミュレーションに追加すると、Houdini はダイナミクスオブジェクトを作成します。ダイナミクスオブジェクトは、オブジェクトのジオメトリを使用して、***密度***、***摩擦***、***跳ね返り***などのダイナミクスプロパティを追加します。



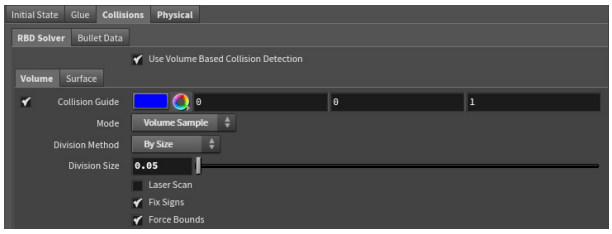
アクティブと Static（静的）

アクティブなダイナミクスオブジェクトは、フォースや衝突の影響を受けますが、Static オブジェクトは影響を受けません。アニメーションジオメトリや変形ジオメトリを使用したい場合は、ダイナミックオブジェクトの **Initial Object Type** メニューまたは **Use Deforming Geometry** チェックボックスを使用して、これを定義する必要があります。

衝突

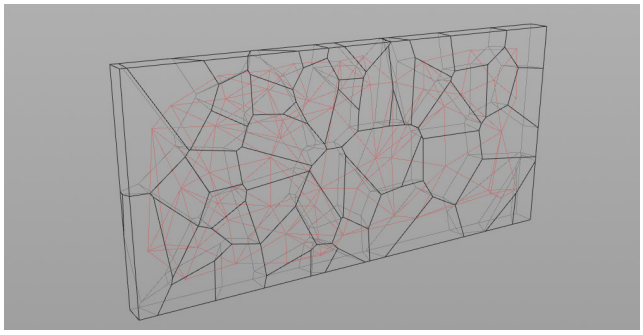
衝突オブジェクトも、シミュレーションに大きく関係する要素です。**地面**をセットアップして、衝突用に連続したサーフェスを作成したり、静的または変形するオブジェクトを使用できます。

それぞれの**ダイナミクスオブジェクト**に、衝突ボリュームを表示したり最適化するための設定があります。衝突の精度はできるだけ高くしたいものですが、それに応じてかかるシミュレーション時間の長さとのバランスも考慮します。



リジッドボディ拘束

Rigid Bodies シェルフには、シミュレーションにも影響させられる拘束が多数あります。例えば、**Pin**、**Spring**、**Slider** などの拘束です。また、リジッドボディシミュレーションのセットアップ時に **Glue Objects** を使用すると、接着を「弱める」か、衝突が発生するまで、オブジェクト同士を接着しておくことができます。

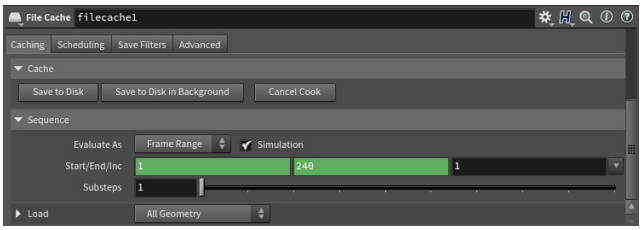


プレイバーの表示

シミュレーションを開始するには、プレイバーで **Play** を押します。シミュレーションが進むと、プレイバーがハイライトされ、どれくらいのシミュレーションがメモリにキャッシュ化されたかが分かります。その後、再度シミュレーションを実行することなく、その領域をスクラブできます。

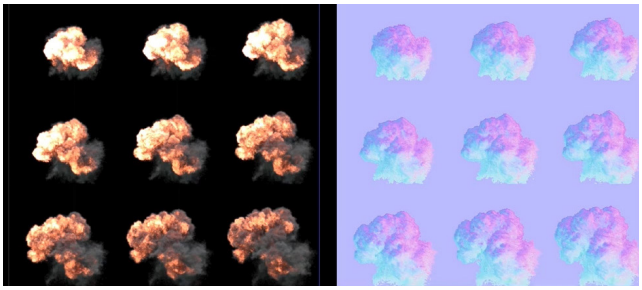
ディスクへのキャッシュ化

シミュレーションが完了したら、DOP 内から **sim** ファイルを保存してロックします。また、**File Cache** ノードを使用して、シミュレーションジオメトリを **bgeo** シーケンスに書き出すことも一般的な方法として使われています。こうすると、プロダクションのライティングやレンダリングの段階で、シミュレーション結果を扱いやすくなります。



ゲーム向けリアルタイム FX

ゲームでは、爆発などのエフェクトをゲームエンジンでリアルタイムに最適化する必要があります。リジッドボディ、Pyro FX、流体など、さまざまな種類の Houdini シミュレーションをゲーム用のアートに変換する方法については、**SideFX Labs ツール**を確認してください。



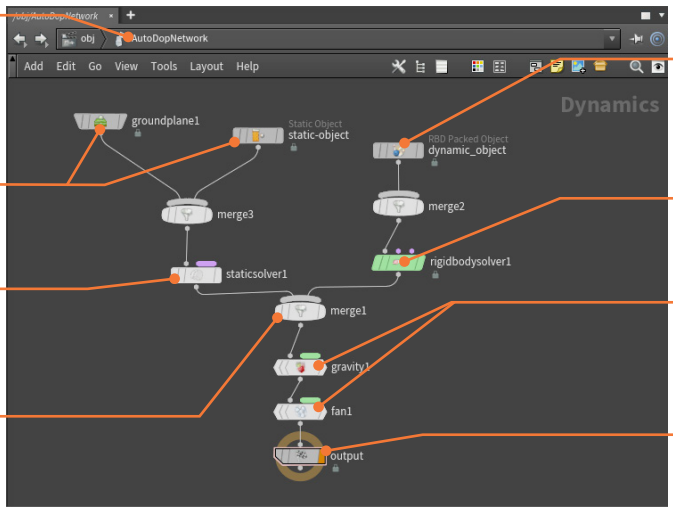
AUTODOPNETWORK

シェルフツールを使用してダイナミクスオブジェクト、衝突オブジェクト、またはフォースを作成すると、全パーツを組み合わせた AutoDopNetwork が作成されます。

Static Object - これらのノードは、地面や静止衝突オブジェクトのプロパティをセットアップします。

Static Solver - ダイナミクスオブジェクトが入力オブジェクトと相互作用する間、入力されるオブジェクトを静止したままにします。

Merge ノード - ダイナミクスシステムの複数の部分を結合します。シミュレーション中、すべてが相互に作用するように、ノードはチェーンの上流と下流で評価されます。



ダイナミクスオブジェクト - このノードはジオメトリを DOP に取り込んで、基本のプロパティを割り当てます。

Rigid Body Solver - 関係するオブジェクトのシミュレーションを生成するソルバです。

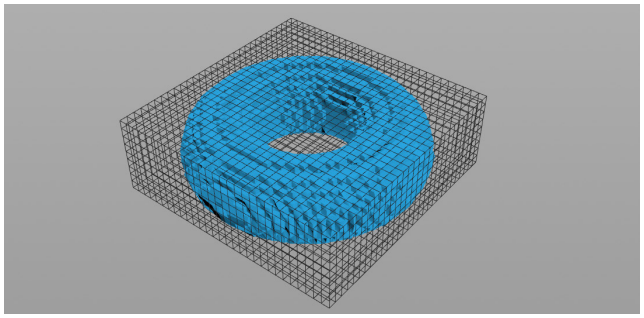
フォース - 重力や風などのフォースを使用して、ダイナミクスオブジェクトに影響を与えるノードです。

Output ノード - このノードを使用すると、シミュレーションをキャッシュ化したい場合に .sim ファイルを出力できます。

Cloud FX とボリューム

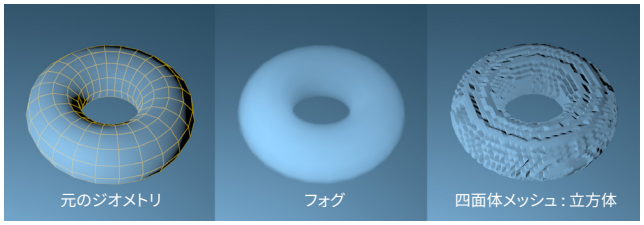
Houdini のビジュアルエフェクトで重要なのは、ボリュームデータの使用です。Houdini では、ボリュームは内部的にツールの機能をサポートしますが、ボリュームが何であるか、さらにはボリュームを直接扱う方法を学ぶことをお勧めします。

ボリュームでは、ポイントやポリゴンではなく、***ボクセル***を使用してオブジェクトを表現します。ボクセルは 3 次元ピクセルです。立方体のグリッドを構成する各ボクセルには、ボリュームの表示方法を示す情報が含まれ、うっすらとした雲のような形状の表現にも適しています。ボリュームベースのオブジェクトの視覚的な品質は、3D グリッドの解像度に応じて決まります。解像度が高いほど、品質は高くなりますが、パフォーマンスに影響があります。



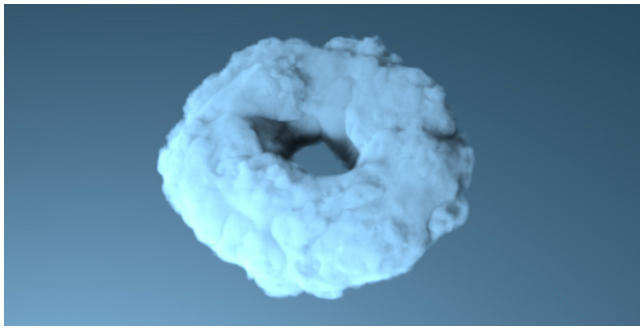
ISO OFFSET

Isooffset ノードはジオメトリコンテキストにあり、マニフォールド（多様体）のポリゴンジオメトリを受け取って、Houdini が使用できる Houdini ボリュームを構築します。さまざまな Output タイプが選択可能で、***フォグ***や***四面体メッシュ***として形状を表示することができます。

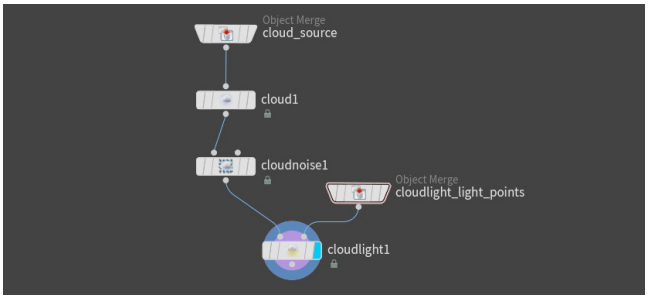


CLOUD FX

このツールセットは、ジオメトリを雲のような VDB ボリュームに変換し、ライティングを施します。**Cloud Rig** ツールは、個々の雲を形成するだけでなく、**Cloud**、**Cloud Noise**、**Cloud Light** など、最終的なルックに寄与する低レベルツールを理解するためにも役立ちます。



構築されたネットワークは、雲のソースを読み込んでから他のノードを適用することで、Houdini ボリュームと VDB による雲のようなエフェクトを作成します。Houdini には、空をボリュームクラウドで埋め尽くす **Sky Rig** ツールも用意されています。



Unreal などのゲームエンジンで使う雲の風景を作成するには、Houdini で Sky Rig を使用し、それをメッシュに変換すれば、生成サーフェスとして使用できます。このアプローチに関しては、SideFX の Web サイトで **Andreas Glad** によるチュートリアルを確認してください。

OPEN VDB

「OpenVDB」は、アカデミー賞を受賞したオープンソースの C++ ライブラリで、3 次元グリッド上に離散化されたスパースボリュームデータを効率的に格納および操作するツール群で構成されています。DreamWorks Animation 社によって開発およびメンテナンスされており、長編映画制作でのボリュームアプリケーションでよく使用されます。- openvdb.org

Houdini には、さまざまな OpenVDB ボリュームノードがあり、ジオメトリをボリュームに変換するジオメトリ（SOP）ネットワークで利用できます。

内部的なボリューム

Houdini のツールの多くは、見えないところで、つまり内部的にボリュームを使用しています。以下は、ボリュームが作業に寄与している例です。

- **コライダ** - デフォルトで、ボリュームはダイナミックシミュレーション用にジオメトリをコライダに変換します。
- **シミュレーションフィールド** - ボリュームは、ダイナミックシミュレーションに寄与する Density（密度）や Velocity（速度）などのフィールドを定義します。
- **ヘアーとファーのツール** - これらのツールは、グルーミングの計算を支援するためにボリュームデータを使用します。
- **地形** - Height Field ツールは、それぞれのボクセルに各グリッドポイントでの地形の高さを含む 2D ボリュームを使用します。
- **レンダリング** - ボリュームは、Mantra で Water Depth（waterdepth）とフォグのエフェクトを作成します。

地形と Height Field

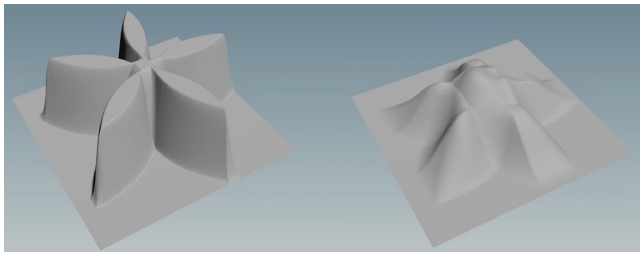
Houdini の一連の heightfield ノードを使用すると、プロシージャルに地形を生成できます。形状を重ねたり、ノイズを追加したり、侵食のシミュレーションを実行することで、デジタル景観のルックを定義できます。このワークフローはコンポジットに似ていますが、すべての作業を 3D 形状で行います。

Houdini には、地形を生成および形成するための各種ジオメトリノードが用意されています。これらのツールは、2D ボリュームを使用して地形を表現し、各ボクセルにはそのグリッドポイントでの地形の高さ、つまり **Height Field** が含まれています。ジオメトリネットワークに渡されるデータには、複数の Height Field を含めることができます。これらのツールには、**Terrain** デスクトップからアクセスできます。Houdini のビューポートでは、2D Height Field を 3D サーフェスとして視覚化でき、マスクフィールドは 3D サーフェス上に赤色で表示されます。Height Field のレンダリング専用の Mantra プロシージャルがあり、ダイナミックシミュレーションでは衝突サーフェスとして使用できます。


ARON KAMOLZ

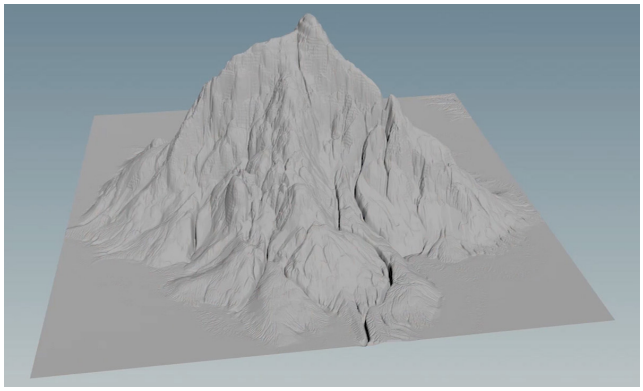
パターン

Heightfield ノードを配置して基本の解像度を定義すると、**Heightfield Pattern** ノードを使ってさまざまな初期形状にアクセスできます。線形、同心円、放射状のランプ、線形階段、星などの放射状に対称的な形状、ボロノイセルなどをセットアップできます。これらの形状をブラーさせたり歪ませて、地形の開始点となる形状を作成します。また、複数の要素を組み合わせたり重ねることで、さらに洗練された結果に仕上げることも可能です。



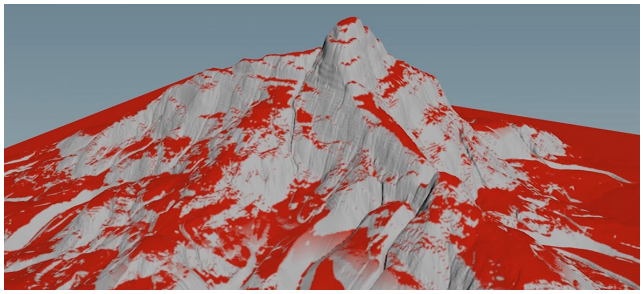
ノイズ

地形を構築する際に、レイヤーにノイズを追加して自然なルックにすることができます。***Perlin***、***Sinusoid***、***Worley*** など、さまざまな種類のノイズが用意されています。地形のリアルさが増すだけでなく、さまざまなノイズを適用した複数の形状を組み合わせることで、多様性に富んだハイパーリアルな結果を得られます。



マスク

Height Field ツールはまた、それぞれのボクセルに**マスクレイヤー**を含む 2 次的なタイプの 2D ボリュームも使用します。ほとんどの Terrain ノードは、2 番目の入力でマスクレイヤーを受け取り、そのノードが変更する地形の領域をマスクによって制御します。マスクの利用方法はさまざまあり、マスクを使用することで、ディテールの追加や地形の形成が行いやすくなります。Height Field にマスクをペイントすることも可能です。



浸食

Heightfield Erode ノードは、雨量、土壌の侵食性、エントレインメント率を変数として使用することで、侵食や堆積をシミュレートします。このノードは、プレイバック中に反復的に動作します。最初のフレームでは何の影響も受けていないように見えます。浸食のシミュレーションを確認するには、プレイバーで Play を押す必要があります。

エクスポートオプション

ゲームエンジンなどの他のアプリケーションで使用するために、地形をエクスポートする方法は 2 通りあります。**Heightfield Output** ノードを使用すると、Height / マスクレイヤーを画像としてディスクにエクスポートし、テクスチャとして取り込むことができます。あるいは、**Houdini デジタルアセット**を作成し、Houdini Engine プラグインを使用して Unreal や Unity などのアプリケーションで開くことも可能です。これらのデジタルアセットは、ゲームエンジンの組み込みの地形ツールで扱うことができます。

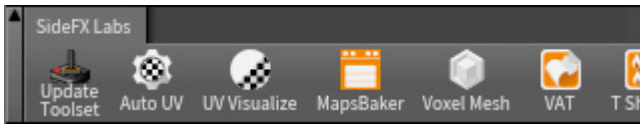
SideFX Labs

SideFX Labs は、Houdini のアーティストおよびゲーム開発関連の向けワークフローの高速化を目的とした、ハイレベルなツール群です。メッシュ処理、リアルタイム FX、UV 編集、シミュレーションからのモーションベクトル作成など、開発中のツールも多数あります。

Houdini のすべての機能は映画、TV、ゲームのコンテンツ制作に使用できますが、Labs ツールセットは、現在の Houdini では対応が難しい、アーティスト固有のタスクにも対応できます。通常の Houdini 開発サイクルとは別に開発されており、テスト目的での試用が可能になった時点で提供されます。Houdini 内から直接ダウンロードするか、SideFX Labs の **GitHub** ページからアクセスできます。

ツールのダウンロード

Labs ツールは、Houdini と一緒にインストールすることも、SideFX Labs シェルフタブからアクセスすることもできます。たいていはデスクトップに表示されないため、追加するための操作が必要です。表示されたら、**Update Toolset** ボタンをクリックします。すると、インストールを促すダイアログが表示されます。ツールの多くがベータ版のため、このときには **Production Builds Only** オプションを**オフ**にします。



シェルフにはツールのほとんどが表示されますが、ビューポートやネットワークビューで **Tab を押した**場合にのみ利用できるツールもあります。こうしたノードは、識別しやすいように先頭に **Labs** が付いています。

FX ツール

Houdini は強力な FX ツールで知られていますが、SideFX Labs には、ゲームまたはバーチャルプロダクションなどのリアルタイム環境で使用するために結果を処理するツールがあります。また、シミュレーションを最適化して、テクスチャ、FBX、CSV などに出力する各種ツールもあります。

Vertex Animation Textures - Vertex Animation Textures ROP は、布、リジッドボディ破壊、流体、パーティクルの複雑なアニメーションを再生する、リアルタイムマテリアルで使用するメッシュとテクスチャをエクスポートします。

Flipbooks Texture - Pyro FX 向けテクスチャアトラスを作成およびプレビューできる、高速 GL または Karma ベースのツールです。

Destruction Cleanup - リジッドボディのシミュレーション結果をエクスポートする前に、余計なジオメトリを削減し、法線を整え、アトリビュートを整理します。

Skinning Converter - Skinning Converter は、トポロジが変わらない変形メッシュシーケンスをボーンベースのアニメーションに変換する SOP です。

Make Loop - アニメートさせたメッシュ、ポイント、ボリュームを受け取ってループにします。

Volume to Texture - Volume Texture ツールを使用すると、Ryan Brucks 氏による UE4 のボリュームプラグインで使用可能なテクスチャを書き出せます。

Flowmap - このユーティリティツールは、入力ジオメトリ上にフローマップテンプレートをセットアップします。

Flowmap Visualize - Houdini ビューポートでのフローマップテクスチャの高品質リアルタイムプレビューです。

Flowmap Obstacle - Flowmap Obstacle SOP を使用すると、障害物ジオメトリに基づいてフローマップを簡単に修正できます。

Niagara ROP - 弾丸シミュレーションから衝撃、分割データ、補間データを抽出および書き出して、UE4 Niagara データインターフェースで使用できるようにするオールインワン HDA です。

Gamedev Procedural Smoke - Procedural Smoke SOP は、煙を表現したアニメーションボリュームを生成します。

ROP Vector Field - ボリュームまたはポイントクラウドから、UE4 と互換性のあるベクトルフィールドを生成します。

メッシュ処理

高精度のメッシュをゲームに取り込むまでには、フォトグラメトリ、トポロジのクリーンアップ、メッシュのリダクション、UV レイアウト、マップのベイクなど、いくつものステップがあります。Labs ツールを使用と、ワークフローをまとめ、一般的なソフトと統合することで、このプロセスを簡略化できます。

AliceVision Photogrammetry - AliceVision は、3D 再構築とカメラ追跡アルゴリズムを提供する、フォトグラメトリのコンピュータビジョンフレームワークです。

ZBrush Bridge - GoZ は Zbrush の高速ファイル転送機能です。ファイルのパスや拡張子といった設定なしに、Houdini と Zbrush 間で、シームレスにメッシュを送信できます。

Delete Small Parts - 接続性とサイズに基づいてパーツを削除します。

Delight - 高解像度のフォトグラメトリスキャンに含まれる、アンビエントライティング情報を除去します。

GameRes - 高解像度モデルを低解像度モデルにする、フルパイプラインノードです。

Maps Baker - ほぼインタラクティブな速さで、高解像度モデルから低解像度モデルへのテクスチャベイクを生成します。

LOD Hierarchy - LOD 階層を作成して FBX としてエクスポートします。

Mesh Sharpen - メッシュの曲率に基づいてジオメトリを鋭くします。

Edge Damage - ジオメトリのエッジに摩耗を加えます。

ワールドの構築

デジタルワールドはより大きく、より複雑になっています。効率的なワールド構築ワークフローを用いることが重要です。ニューヨークを再現したい、密林を育てたい、SF アドベンチャーに屋内のディテールを加えたい。このようなときが、Lab ツールの出番です。

Physics Painter - Physics Painter は、ユーザが他のオブジェクト上に物理オブジェクトをペイントできるようにする SOP です。

Building Generator - ユーザ定義モジュールのライブラリを使用して、低解像度のブロックアウト（枠組み）ジオメトリを詳細なビルに変換します。

OSM Import - Open Street Map は、街路データの優れたデータベースです。このノードは、OSM ファイルや、ビルや道路のさまざまなタグ付きアトリビュートをすべて効率的に Houdini にロードします。

OSM Buildings - OSM データからビルを生成します。

Tree ツール - Labs の Tree ツールにはいくつかのツールがあり、組み合わせて使用することで木、茂み、植物などの複雑な枝構造を作成できます。

Cable Generator - ケーブルの高い位置の「ピン」ポイントと低い位置の「垂れ」ポイントを表現したカーブを与えると、この SOP はユーザ定義可能なケーブルの本数、形状、色で垂れ下がったケーブルを生成します。

Curve Branches - カーブ上にカーブを散乱させます。いくつもの直感的なコントロールを使用して、クリーンな幾何学的な枝から有機的なブドウの枝までを表現可能です。この SOP を複製してチェーン接続すれば、再帰的成長を表現でき、L-System のようなルックをより柔軟な制御で得ることができます。

Dirt Skirt - オブジェクトと地面が交差した箇所にジオメトリ「スカート」を作成します。これは、ゲームエンジンでソフトブレンドとして使用します。

Lot Subdivision - ポリゴンをパネルに分割します。市街地の作成やディティールアップに便利です。

MapBox - mapbox.com で提供されているデータを使用して、色、高さが定義された Open Street Map（OSM）カーブを生成します。

SciFi Panels - SF 風のパネルを生成するサンプル HDA です。

Snow Buildup - 入力メッシュに積雪を模倣したジオメトリを追加します。

Terrain Texture Output - Terrain Texture ROP SOP は、Height Field から画像データをレンダリングします。

モデリング

Labs ツールには、ゲーム用のジオメトリを簡単に作成できるように設計された、さまざまモデリングツールがあります。

Decal Projector - デカール（局所的なジオメトリとテクスチャ）をジオメトリ上に投影します。

Calculate Slope - 方向と比較してサーフェスの勾配を計算し、オプションでその結果をブラーしたり再マップすることができます。

Curve Sweep - プロファイルのタイプ、幅、捻じりの挙動の単純なコントロールを使って、入力カーブに沿ってプロファイルを回転させます。

Extract Silhouette - XYZ のいずれかの軸から投影されたオブジェクトのアウトラインを作成します。

UV マッピング

テクスチャ UV は、ゲームアート作成で大きい割合を占める作業工程です。これらのツールは Houdini の既存の UV ツールセットを強化するもので、より素早く効率的に作業できるようにします。

Auto UV - 自動的にオブジェクトにシーム（継ぎ目）を生成し、UV Flatten を即時に実行します。

Inside Face UVs - 破壊されたジオメトリの内部フェースに UV を生成します。

UV Transfer - ソースジオメトリとターゲットジオメトリの間で UV を転送します。

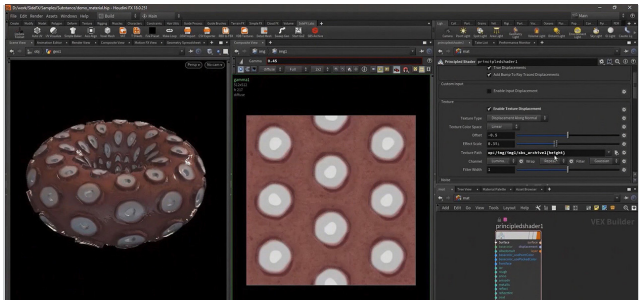
UV Visualize - UV を視覚化するヘルパースクリプトです。シーム（継ぎ目）の視覚化、UV 空間とモデリング空間の間のワープ、グリッドテクスチャのタイリングの変更、島の視覚化といった機能が含まれます。

Texel Density - このツールは、アセットとプロジェクトの解像度に基づいて、プリミティブ単位でアセットの現行テクセル密度を計算します。

インテグレーション

これらのツールは、Houdini とゲームエンジンの間でのインポート・エクスポートが簡単になります。

Substance COP - Houdini 向け Substance プラグインを使用すると、Substance Archive ファイルを Houdini の COP に読み込むことができます。



Rizom UV - RizomUV Bridge は、4 種類の SOP のセットで、Houdini と RizomUV 間のやり取りを簡易化します。

Quad Remesher - QuadRemesher ノードは、Exoside 社の QuadRemesher コマンドラインインターフェースのラッパーです。

Instant Meshes - DDS（DirectDraw Surface）ファイルを読み込みます。

Sketchfab - ジオメトリを Sketchfab にアップロードします。

3D Facebook Image - 3D シーンを Facebook にアップロード可能な 2.5D 画像に手早く出力できます。

Marmoset ROP - Marmoset ROP を使うと、Houdini 内で手軽に mview を生成できます。

Gaea Tor Processor - Gaea Tor Processor を使用して、Gaea で作成されたビルドの .TOR ファイルをロードできます。

UX

Labs ツールの中には、Houdini を使用するアーティストのユーザエクスペリエンスの向上を目的に作られたものもあります。

Crash Recovery - File メニューにあるこの機能は、不運にもクラッシュしてしまったファイルを素早く修復します。

Network Paint - ネットワークエディタで描くだけで、カラフルな注釈をネットワークに追加できます。

Sticker Placer - 数字、アイコン、ユーザが作成したグラフィックスを配置して、ネットワークに注釈を付けることができます。

External Script Editor - VEX、Python、OpenCL、エクスプレッションを扱う際に、外部 IDE とのライブ接続をセットアップできます。

その他

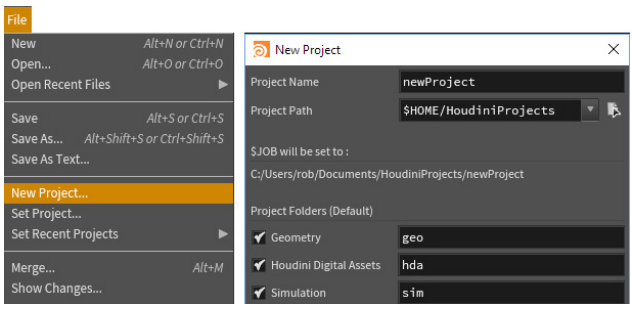
他にもさまざまなツールが追加されています。***SideFX.com/labs*** をご覧ください。

ファイル管理

アーティストとして成功するには、Houdini での作業で作成したすべてのファイルの管理方法を理解することは、とても大切です。一般的なシーンファイルは、ディスク上で外部依存関係を持っていることがあります。ファイルを別のコンピュータに移動する場合には特に、これらの管理が重要になってきます。

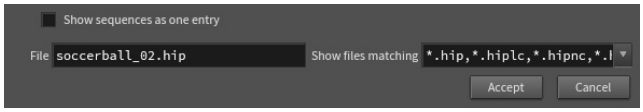
プロジェクトディレクトリ

Houdini はハードドライブ上に散在するファイルを扱うことはできます。そうすると、作業内容を共有したり、ファイルの依存関係を管理するのが難しくなってしまいます。**File > New Project** を使用してプロジェクトディレクトリをセットアップするか、**File > Set Project** を使用して既存のプロジェクトディレクトリを作業の「ホームベース」として選択しましょう。こうすると、必要なプロジェクトファイルすべてに、ローカルの依存関係が簡単にセットアップできます。



シーンファイル | .hip

Houdini で主に使用するファイルタイプは、**.hip** ファイルです。このファイルには、すべてのノードとネットワークが含まれています。作業内容を保存する際は、このファイルタイプを使用します。

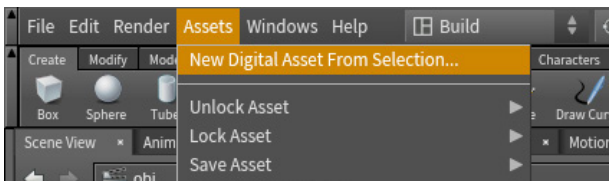


UNIVERSAL SCENE DESCRIPTION | USD

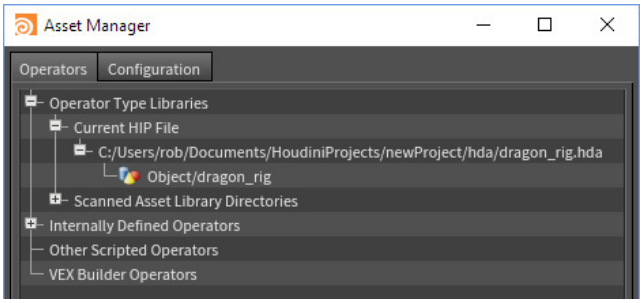
Houdini では、Solaris のライティングとルックデブ環境は、PIXAR が作成したオープンソースイニシアティブである USD（**Universal Scene Description**）を使用しています。Solaris では USD がネイティブであり、プロシージャルノードを使用して、参照、ペイロード、レイヤー、コレクション、バリアント、詳細レベルを管理できます。

Houdini デジタルアセット | .hda

Houdini ネットワークをカプセル化し、**Houdini デジタルアセット**つまり **.hda** ファイルに保存することも可能です。アセット内のパラメータをトップレベルにプロモートして、アセット用のカスタム UI を作成できます。これらのファイルは他のアーティストと簡単に共有でき、プロジェクトのライフサイクルを通じてアセットが発展していく間も、堅牢な参照アーキテクチャを提供します。

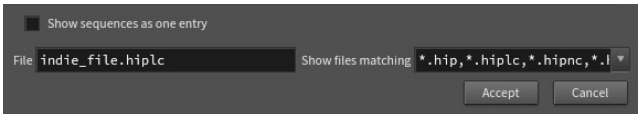


アセットの作成と読み込みには、**Asset** メニューを使用します。また、シーンに読み込まれたアセットは、そのメニューにある **Asset Manager** を使用して管理できます。シーンに読み込まれた 2 つの HDA ファイルが同じ名前の場合、Houdini はマネージャで設定されたルールに基づいてどちらかを選択します。HDA ファイル内のアセット定義に加えられた変更は、そのファイルを参照するシーンに自動的に反映されます。なお、古いデジタルアセットファイルは拡張子が .otl の場合がありますが、機能は .hda ファイルとまったく同じです。



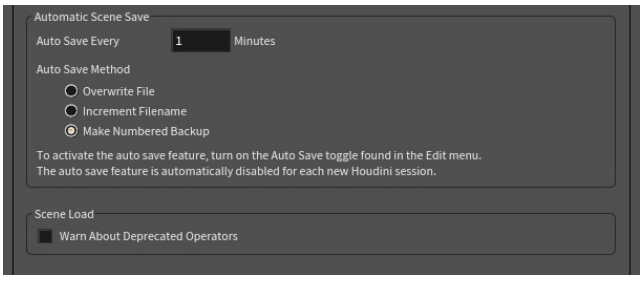
APPRENTICE と INDIE ファイル

Houdini Apprentice と **Houdini Indie** は、商用バージョンの Houdini では開くことができない、別のファイルタイプを使用します。Apprentice はシーンに .hipnc（non-commercial：非商用）、アセットに **.hdanc** ファイルを使用し、Indie はシーンに **.hiplc**（limited commercial：限定的な商用）、アセットに **.hdalc** ファイルを使用します。



作業のバックアップ

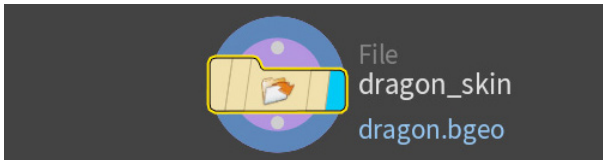
デフォルトでは、Houdini は保存するたびに、シーンファイルとデジタルアセットファイルの番号付きのバックアップを作成します。前のイテレーションを見直したい場合や、作業ファイルに問題があった場合は、そのファイルに戻ることができます。また、**Edit > Preferences > Save and Load Options** で Houdini を **AutoSave** に設定することも可能です。なお、これらのバックアップファイルはディスク容量を消費するため、時々削除しましょう。



FILE SOP

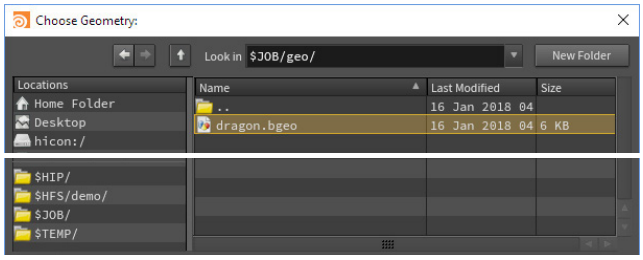
File > Import > Geometry を使用してジオメトリを Houdini にインポートすると、ジオメトリ（SOP）レベルに **File** ノードが配置されます。このファイルはディスク上のファイルとの接続を維持し、ファイルに

変更が加えられると、Houdini シーンも更新されます。接続を解除したい場合は、File ノードをロックする必要があります。



ファイルの依存関係（$HIP/$JOB）

ジオメトリやテクスチャファイルなど、ディスク上のファイルを参照するノードを使用している場合には、プロジェクトディレクトリを別のコンピュータやクラウドに移動した場合にどうなるかは、パスによって決まります。絶対パスは、ファイルを移動すると切れてしまいます。したがって、シーンファイルをパスの「ホームベース」として使用する **$HIP** や、プロジェクトディレクトリを使用する **$JOB** を使用する必要があります。シーンファイルが適切にセットアップされているかどうかを確認するには、**Render > Preflight Scene** を使用します。



ディスク容量の管理

巨大なシーンファイル、バックアップファイル、大規模なシミュレーションは、**かなりのディスク容量**を消費します。ディスクの容量を使いすぎて、コンピュータが不安定にならないように注意してください。容量が大きいファイルは外部ドライブに保存し、コンピュータのメインディスクには日常業務をこなすのに十分な容量を残しておきます。

相互運用性

Houdini からインポートしたり、エクスポートする際に使用できるファイルフォーマットは、多数あります。一般的な Houdini パイプラインで使用される主なフォーマットを紹介しましょう。

Houdini ファイル - Houdini でのみサポートされるファイルフォーマットをいくつか紹介します（.hip と .hda を除く）。

.bgeo - ジオメトリと UV、Velocity、法線などの関連アトリビュートを保存するフォーマットです。アニメーションやシミュレーションを番号付きの bgeo ファイルとして保存することで、動きも保存可能です。bgeo.gz ファイルは、このフォーマットの圧縮バージョンです。

.sim - シミュレーションデータを保存して、シミュレーションをディスクにキャッシュ化することができます。このファイルも使用できますし、.bgeo でシミュレーションをキャッシュ化することも可能です。

.ifd - Mantra でレンダリングする際に作成されるシーン記述フォーマットです。通常、このファイルは Houdini でレンダリングする際に作成されますが、ディスクに保存して Mantra で直接レンダリングすることもあります。

.pic - Houdini で使用されていた古い画像ファイルフォーマットです。デフォルトのフォーマットは、オープンソースの EXR に置き換えられました。

.rat - この画像フォーマットは、Mantra でレンダリングされるテクスチャマップに最適です。すべてのテクスチャはいずれこのフォーマットに変換されるので、MPlay を使ってこのフォーマットに変換すると、レンダリングが高速化します。

画像フォーマット - ショットのレンダリングやテクスチャマップには、これらの業界標準のフォーマットが使用されます。

.exr - OpenEXR は、Industrial Light & Magic 社が開発したハイダイナミックレンジ（HDR）画像ファイルフォーマットです。Houdini のレンダリングを保存する、デフォルトのフォーマットです。

.jpg/.png - 画像を Web にパブリッシュする際に使用されるフォーマットです。
.tga/.tif - ビデオゲームのテクスチャマップによく使用されるフォーマットです。

ジオメトリフォーマット - ジオメトリのインポートとエクスポートでは、次のフォーマットが最も一般的です。

.usd - Solaris/LOP で使用されるフォーマットで、他のアプリケーションと共有するためのオープンソース交換フォーマットでもあります。

.abc - Alembic は、オープンコンピュータグラフィックス交換フレームワークです。

.fbx - Autodesk 社が所有するフォーマットで、ゲームエンジンや他の 3D アプリケーションとデータをやり取りする際によく使用されます。ジオメトリ、リギング、モーションおよびシェーダ情報を保持できます。

.obj - Wavefront 社が独自に開発したシンプルなジオメトリフォーマットです。

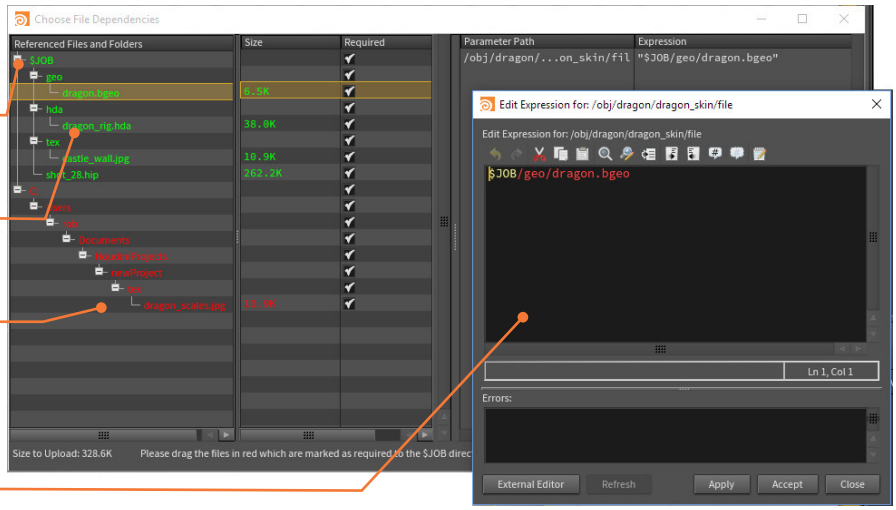
プリフライトパネル
Render メニューから **Pre-Flight Scene** を選択して、シーンのセットアップを評価できます。

参照ファイル - プリフライトパネルでは、$HIP または $JOB のどちらかを参照して、シーンファイルのファイル参照を確認することができます。

緑でハイライトされた参照 - $HIP や $JOB に対する相対参照は緑で表示され、正常に機能していることが示されます。

不正確な参照 - ファイル参照が絶対パスであり、$HIP や $JOB に対する相対参照でない場合は、赤で表示されます。プロジェクトを他のアーティストと共有したり、クラウドで共有する場合は、事前に修正する必要があります。

Edit Expression - 任意のファイル名をクリックし、エクスプレッションを右クリックすると、Edit Expression ウィンドウが開きます。

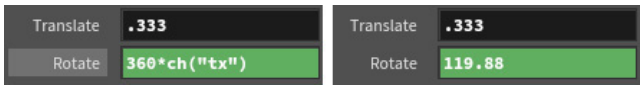


エクスプレッションとスクリプト

Houdini はプロダクションレベルのソリューションであるため、作業ではスクリプトが重要な役割を果たします。アーティストはエクスプレッションを書くだけで十分なことがほとんどである一方、テクニカルディレクターはより長い時間をこうしたテクニックに費やすことになります。Houdini は、Hscript、Python、VEX をサポートしています。

HSCRIPT エクスプレッション

HScript は、エクスプレッションの記述に使用できる情報を取得および操作するための、素早く簡潔な方法となるように設計されています。エクスプレッションとは通常、単純な文字列や数値ではない、任意の値です。これは、変数のような単純な場合もあれば、方程式やエクスプレッション関数である場合もあります。



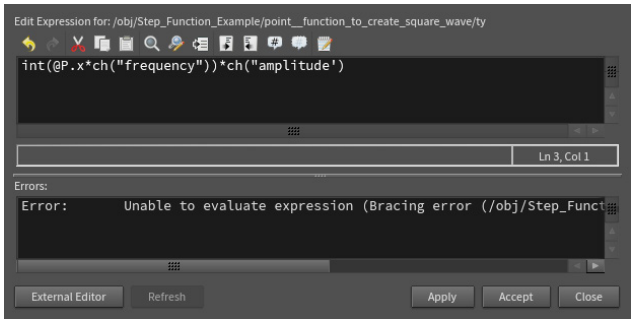
フィールドにタイプするだけで、パラメータに直接エクスプレッションを入力することができます。**Enter** を押すと、フィールドが緑でハイライトされます。パラメータ名をクリックするだけで、エクスプレッションとエクスプレッションの結果を切り替えられます。

チャンネル参照を作成する場合は、パラメータを **RMB クリック**して **Copy Parameter** を選択し、リンクさせたいパラメータに移動して **Paste Relative References** を選択します。

また、2 つ目のパラメータを RMB クリックし、**Reference > Scene Data** を選択しても同じです。表示されるパネルで、他のオブジェクトやノードからデータを選択すると、エクスプレッションが構築されます。この方法で、複数のパラメータにエクスプレッションを設定することも可能です。

エクスプレッションエディタ

関数の複雑さやパラメータの種類によっては、**エクスプレッションエディタ**を使用することもできます。エクスプレッションエディタを開くには、パラメータを **RMB クリック**して **Expression > Edit Expression** を選択するか、パラメータ上にマウスを置いて **Alt + E** を押します。



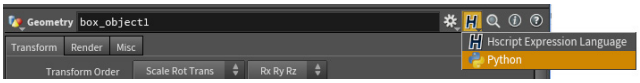
PYTHON

Python は、CG 業界では統合や標準化をサポートするスクリプト言語として一般に使われ、よく知られています。その理由から、ツール開発に向いています。

Houdini における Python は、Houdini Object Model（HOM）上に構築されています。HOM は、Python スクリプト言語を使用して Houdini から情報を取得し、制御することができる API です。

Python では、HOM を定義するモジュール、関数、クラスの階層のトップが hou パッケージです。hou モジュールは、パラメータエディタや hython コマンドラインシェルでエクスプレッションを記述するときに自動的にインポートされます。

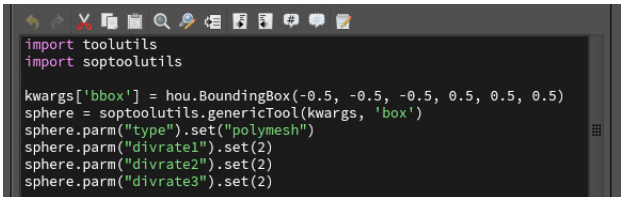
Houdini でエクスプレッションを記述するのに Python を使用することもできます。これには、ノードのパラメータエディタの上部にあるエクスプレッション言語オプションを変更します。



Python コマンドの入力には、**Python シェルパネル**も使用可能です。また、hou モジュールを標準の Python シェルにインポートして、Houdini を既存の Python ベースのスクリプトに統合することもできます。

ツールシェルフ

シェルフツールは Python でセットアップされています。こうしたコードを確認するには、任意のシェルフツールを **RMB クリック**して **Edit ツール**を選択します。

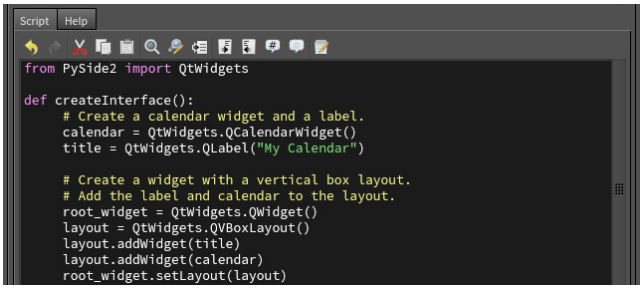


```
import toolutils
import soptoolutils

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

Python Panel エディタペインでは、PySide2 または PyQt5 インターフェースを作成、編集、削除できます。また、このエディタでは、Houdini ペインタブメニューのエントリだけでなく、Python Panel のインターフェースメニューのエントリも管理できます。このパネルにはサンプルコードがいくつか付属しており、試してみることができます。



```
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 ステート

Python で Viewer State を記述すると、ビューポート内でのノードに対するユーザ操作をカスタマイズできます。これを利用して、アーティストフレンドリーなインターフェースをツールに構築できます。詳細な情報は、ドキュメントを参照してください。

VEX

VEX はハイパフォーマンスなエクスプレッション言語で、シェーダの記述など、Houdini のほとんどの場所で使われています。VEX 評価は、一般に非常に効率的で、コンパイル済み C/C++ コードに近いパフォーマンスを発揮します。

VEX はスクリプトの代用ではありませんが、シェーダやカスタムノードを記述するために使う、小さくて効率的な汎用言語です。VEX は大まかには C 言語ベースですが、RenderMan シェーディング言語と同様に C++ の考え方を採用しています。

VEX は、Houdini のさまざまな場所で使われています。

モデリング – VEX SOP により、Point アトリビュートを操作するカスタムサーフェスノードを記述できます。ポイントを動かしたり、速度を調整したり、色を変更することができます。さらに、ポイントをグループ化するなど、さまざまな便利なタスクを実行できます。

レンダリング – Karma と Mantra が、シェーディング計算に VEX を使用しています。これには、ライトシェーダ、サーフェスシェーダ、ディスプレイスメントシェーダ、フォグシェーダが含まれます。

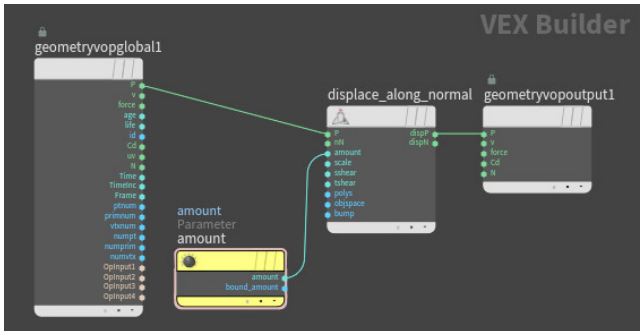
コンポジット – VEX Generator および VEX Filter COP を使用すると、複雑なカスタム COP を VEX で記述できます。エクスプレッションは C/C++ と同様の速度で評価し、Pixel Expression COP よりも 1000 倍高速に実行されます。

CHOP– VEX CHOP で CHOP をカスタマイズすることができます。CHOP 関数は任意の数の入力チャンネルを操作して、任意の方法でチャンネルデータを処理できます。場合によっては、VEX コードはコンパイル済み C++ コードよりも高速で実行可能です。

ファー – プロシージャルなファーの動作を VEX で実装しています。

VOP

VEX を使いたいが、コードは書きたくない場合には、VOP コンテキストでノードベースのインターフェースを使用することができます。このためには SOP コンテキストで **Attribute VOP** ノードを使用し、中に入って VOP を使って VEX コードを作成します。入力ジオメトリを受け取って、それを操作できます。

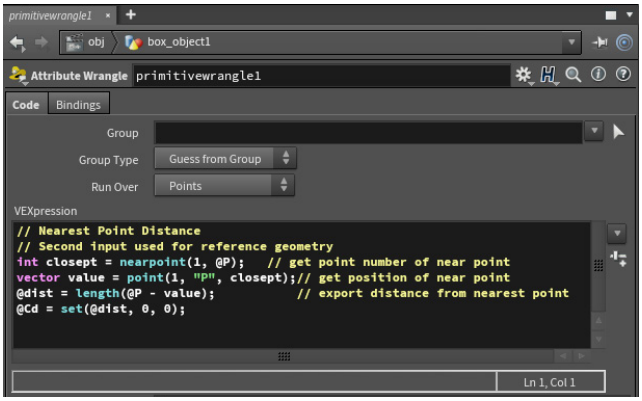


Parameter VOP を使用すると、SOP レベルで使用する、浮動小数点のスライダのようなインターフェース要素を構築できます。こうすると、VOP レベルに戻ることなく VEX コードを実行できます。

VOP コンテキストは、アーティストがインタラクティブに VEX コードを作成できるように設計されています。スクリプト経験者にとっては、Wrangle ノードに直接コードを記述する方が合理的かもしれません。

WRANGLE ノード

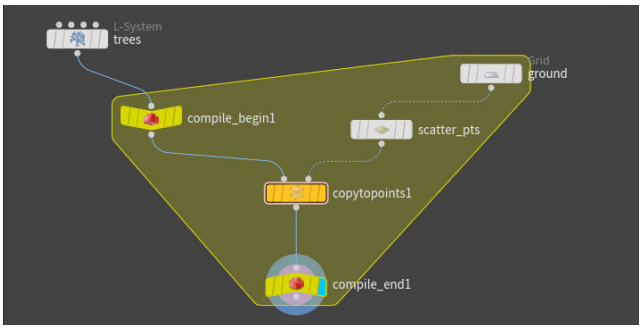
VEX を使い慣れたユーザがアトリビュートを微調整する低レベルの **Attribute Wrangle** ノードを含む、Wrangle ノードを使用することもできます。チャンネル、ボリューム、変形を扱う Wrangle ノードもあります。



Wrangle ノードの使用方法を知りたい場合は、**Entagma.com** をご覧ください。コンテンツを作成する、より技術的なアプローチを学習できます。アーティストの考え方も取り入れた、素晴らしいチュートリアルがいくつも提供されています。

コンパイルブロック

ジオメトリネットワーク(SOP)では、ネットワークの一部をコンパイルブロックに含めることで、コードを記述した場合と同じくらい効率的に機能させることができます。ネットワークの働きにはいくらか制約が課されるとはいえ、適切な状況で使うと、大きなメリットを得られます。



HOUDINI DEVELOPMENT KIT | HDK

Houdini をさらに深く活用するには、HDK があります。HDK は、SideFX のプログラマーが Houdini ファミリー製品の開発に使用しているのと同じ C++ ライブラリの包括的なセットです。HDK を使用すると、Houdini インターフェ-スのさまざまな領域をカスタマイズできるプラグインを作成できます。ここでは、開発キットの用途の例をいくつか紹介します。

- カスタムのエクスプレッション関数を追加する
- カスタムのコマンドを追加する(hscript または HOM)
- カスタムのオペレータを追加する(SOP、COP、DOP、VOP、ROP、CHOP、Object)
- 非標準のレンダラをサポートする出力ノードを追加する
- カスタムのライティングや大気効果をレンダラに追加する

HDK の詳しい使い方は、SideFX の Web サイトにアクセスし、**Support > Documentation > HDK** を選択してください。

タスク

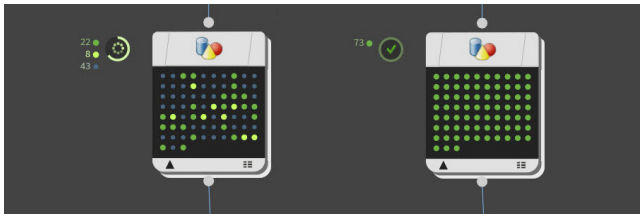
Task Operator（TOP）を使用すると、タスクを整理およびスケジューリングし、インテリジェントに演算ファームに分散できます。これにより、各タスクとタスクの進行状況との関連を示すディペンデンシーグラフを維持しながら、データを並列で処理することができます。

Procedural Dependency Graph

TOP は、Procedural Dependency Graph を使用して構築された Houdini のネットワークタイプです。Procedural Dependency Graph とは、複雑な依存関係をノードで視覚的に記述するテクノロジで、実行可能なタスクセットを生成し、スケジューラの助けを借りて演算ファームに分散させます。結果を評価したら、ネットワーク全体を再クックしなくても、グラフの一部に変更を加えることが可能です。

TOP ノード

タスクまたは TOP ノードを使用すると、パイプラインのタスクを管理できます。各タスクの並列処理と分散が最終的な目的です。TOP ノードがタスクを生成すると、そのタスクはドットで表示されます。タスクがクックされると、このノードや子 TOP ノードで新しいタスクが実行されます。



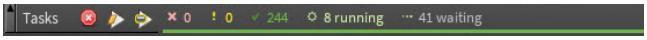
スケジューラ

スケジューラ系ノードは、必要な依存関係を満たしたタスクを受け取り、計算リソースを割り当てます。各タスクが完了すると、スケジューラはタスクグラフに通知します。タスクグラフが PDG グラフに通知すると、次の対象タスクに移行します。PDG は、HQueue、Deadline、Tractor などの業界標準のスケジューラや、Python で記述したスケジューラをサポートしています。

TOP ノードのクック

タスクグラフができたら、ノードをクックします。グラフの途中でノードをクックすることも、チェーンの終端の出力ノードをクックすることも可能です。

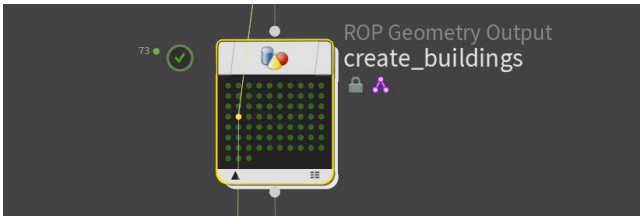
- **Cook Selected Node** Shift + G
- **Dirty and Cook Selected Node** Shift + V



進捗状況は、タスクバーで確認できます。TOP ノードでは、タスクのドットを RMB クリックして、そのタスクをクックするか Dirty（変更あり）にするかを選択できます。タスクを Dirty（変更あり）にすることとは、ネットワークを再クックすると、それらのタスクが再計算されることを意味します。クリーンなタスクは再クックされません。完了した作業をやり直す必要がないことは、TOP のメリットの 1 つです。

依存関係

グラフ内のタスクのドットをクリックすると、そのタスクが依存している上流のタスクと、そのタスクに依存している下流のタスクにつながっている細いラインが表示されます。
上流で変更があった場合、タスクは自動的に Dirty（変更あり）になり、依存関係がある下流のタスクも Dirty（変更あり）になります。PDG グラフが効果的なパイプラインツールとして機能するための、重要なプロセスです。



TOP ノード

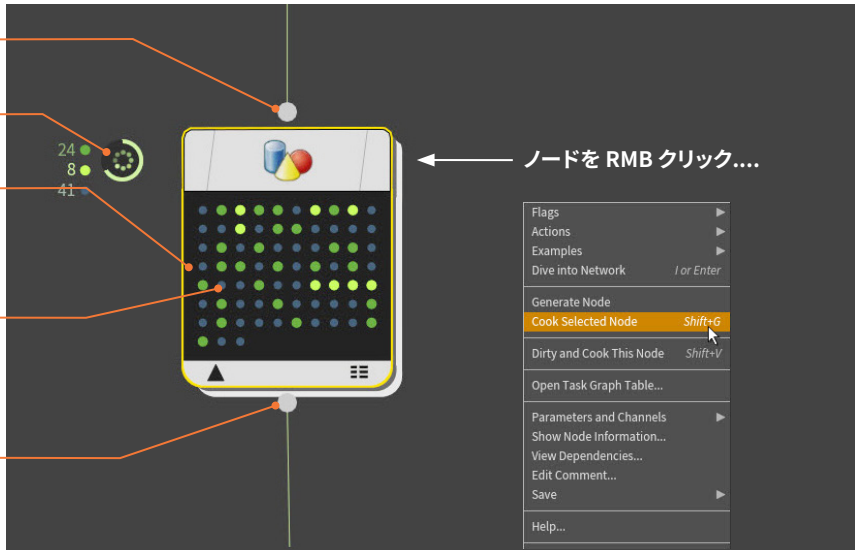
入力 - ノードは、入力に供給される情報を受け取り、データごとに 1 つのタスクに分割します。

進捗ホイール - 完了、クック中、未開始（キュー）のタスクの数を表示します。

TOP ノード - これは、現在クック中のノードです。実行されているタスクに対する指示を含みます。ノードを RMB クリックすると、サポートされるアクションのメニューが表示されます。

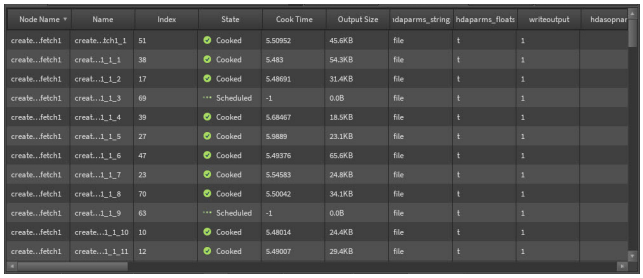
タスク - 各タスクは小さいドットとして示されます。色は現在の状態を示し、タスクのドットを RMB クリックすると、グラフのその部分について詳細を確認することができます。

出力 - タスクが完了すると、このノード上の他のタスクがまだアクティブであっても、出力は次のノードに結果を渡します。



Task Graph Table

ノードを **RMB クリック**すると、**Open Task Graph Table** を選択できます。Task Graph Table には、タスクが項目別にリストされ、インデックス、状態、クック時間、優先度などの情報も確認できます。このウィンドウの項目をクリックすると、ネットワークビューではノード上のタスクのドットがハイライトされます。

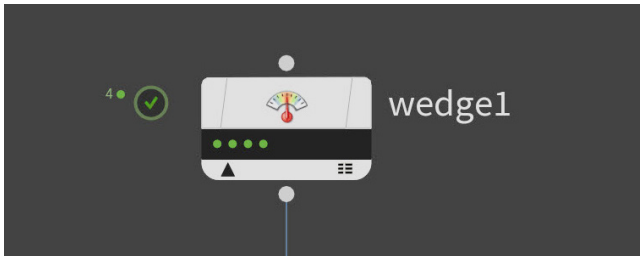


データのインポート / エクスポート

TOP グラフにデータを取り込む際は、さまざまなオプションを使用して、ジオメトリ、画像、スクリプトなどのデータにアクセスできます。Houdini デジタルアセットを使用してプロシージャルネットワークを適用したり、Houdini の他の部分と接続してデータをインポートしたりエクスポートすることが可能です。

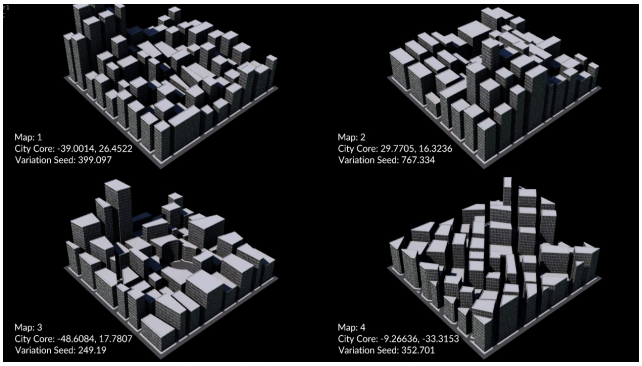
WEDGE ノード

PDG の重要なワークフローである Wedge 化により、デザインの複数のイテレーションを素早く作成できます。その後、すべての異なるオプションを TOP グラフで処理し、最後に収集して最終的な出力を作成できます。



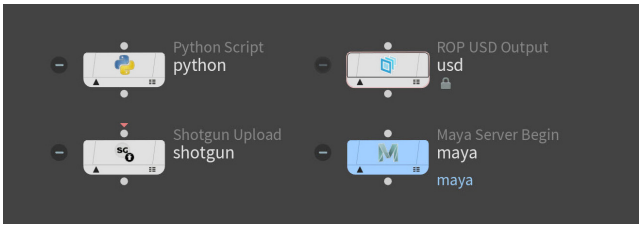
画像モザイクと動画の出力

TOP では、**ImageMagik** を使用してコンタクトシートを作成できます。コンタクトシートを使用すると、デザインのイテレーションを評価して最適な選択をしたり、シーンを彩るプロップのバリエーションを生成することができます。また、オーバーレイを使用してネットワークから情報を引き出すことで、最善の判断を下しやすくなります。



他のアプリとの統合

TOP には、Shotgun や Autodesk Maya などの他のアプリケーションと連携するためのノードが含まれています。ネットワークを Houdini 外に拡張し、パイプラインのすべての部分を補強できます。



Pilot PDG アプリケーション

TOP ネットワークは、Houdini 内からセットアップおよび実行できますが、ファームの管理者や、TOP ネットワークの作成に特化したパイプライン TD は **PilotPDG** を使用できます。Houdini に関連するタスクについては、非グラフィカルで動作する Houdini Engine を呼び出して実行します。

TOP ネットワーク

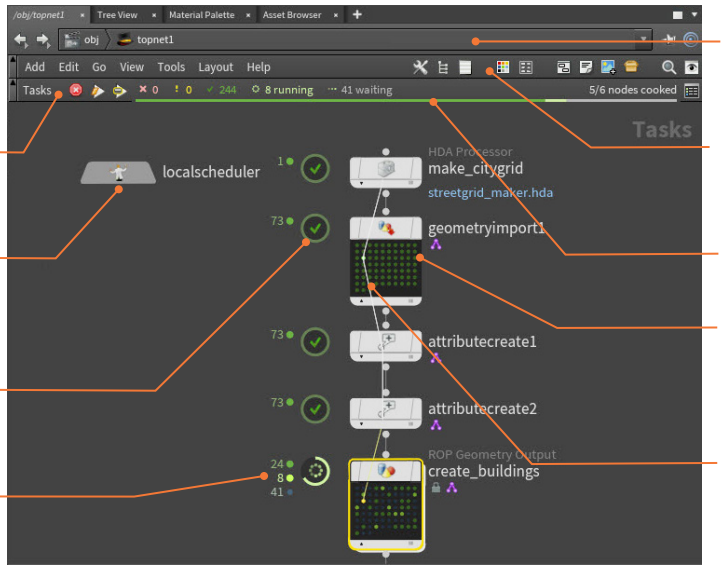
このネットワークタイプでは、処理されるネットワークを管理および確認できます。

タスクバー - ネットワークを開始および停止したり、進捗を確認できます。

スケジューラ - スケジューラ系ノードは、データがどこで処理され、いくつのノードが関与するかを決めます。

完了したタスク - ノードがすべてのタスクの処理を完了すると、チェックマークが付加されます。

進行中のタスク - 進行中には、まだ完了していないタスクを確認できます。



ネットワークパス - グラフがセットアップされている TOP ネットワークまでのパスが表示されます。

TOP メニュー - TOP ネットワークを整理したり処理するための各種オプションがあります。

進捗バー - ネットワークタスク全体の進捗を確認できます。

TOP ノード - これらのノードは、特定のコマンドをタスクに変換し、スケジューラで送信することでタスクを完了します。

依存関係ライン - タスクをクリックすると、ネットワーク上の他のタスクとの接続を確認できます。

HOUDINI デジタルアセット プロシージャルなツール構築

ノードネットワークは、Houdini にプロシージャルな性質を与え、繰り返し適用可能なレシピを定義します。Houdini デジタルアセットを使用すると、これらのネットワークをラップして、カスタムツールやスマートアセットを作成できます。このようにしてアーティストが構築したツールは、何度でも繰り返し使用でき、スタジオ全体の生産性を向上します。

Houdini のノードベースのワークフローが優れている点の 1 つは、アーティストが同じ手順を繰り返すのではなく、既存のノードネットワークに変更を加えることで複数のイテレーションを生成できることです。プロセス全体をはじめからやり直すことなく、いくつものユニークな結果を得られます。

Houdini デジタルアセットはこれをさらに一歩進め、1 つまはた複数のネットワークを単一のノードにカプセル化し、パラメータをトップレベルにプロモートします。このノードはディスクに保存され、作成されたファイルは共有可能で、他のアーティストがそれぞれのシーンにロードできます。

アーティストが構築するツール

Houdini デジタルアセットは、Houdini のインタラクティブなツールを使用して作成します。ノードからアセットプロパティパネルにパラメータをドラッグすることで、ハイレベルなインターフェースを構築でき、コードを記述することなく、カスタムツールを作成できます。つまり、テクニカルアーティストたちはカスタムツールを構築し、それを手軽に同僚に使ってもらうことができるわけです。

Houdini デジタルアセットは、階段や家具などのプロシージャルなプロップや、爆発などのビジュアルエフェクトの場合もあれば、サーフェス上にオブジェクトをばら撒くポピュレートツールなどの汎用的なツールである場合もあります。現在のプロジェクト専用のコンテンツを作成するにしても、すべてのプロジェクトに使える大規模なツールセットを構築するにしても、それぞれの制作ニーズに応じた Houdini デジタルアセットのコレクションを構築できます。

パイプラインフレンドリー

Houdini デジタルアセットがシーンファイルにロードされると、ディスク上の **.hda** ファイルを参照します。つまり、アセットに加えられた変更は、そのファイルを参照しているすべての人に自動的に取得されます。このため、パイプライン全体で、変更を反映するのが非常に簡単です。ディスク上の 1 つのアセットを参照するだけで、最新のイテレーションで更新されたら、即座に新しいアセットにアクセスできるわけです。

Houdini デジタルアセットファイルが格納できるのはアセット定義だけではありません。アセットで使用される画像、ジオメトリファイル、スクリプトも保存できます。したがって、他の人がアセットを使用する際は、関連するすべての情報を利用できます。

CONTENT LIBRARY と ORBOLT

Content Library は、完全なシーンファイル、完全にリギングされたプロップ、レンダリング可能なビジュアルエフェクト、アニメート可能なキャラクタ、ゲームアセットなど、2D および 3D アセットを備えたオンラインアセットリポジトリです。アクセスするには、SideFX の Web サイトで **Get > Content Library** を選択します。

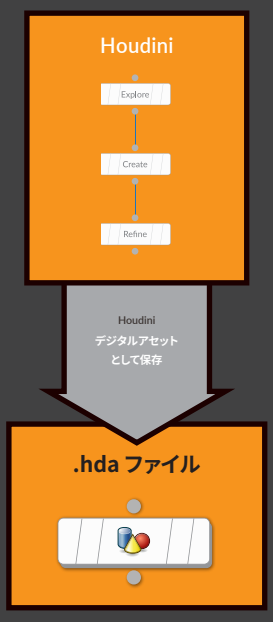
Orbolt は、多種多様なデジタルアセットを提供する、オンラインのアセットマーケットプレイスです。Houdini には、作業時に使えるように、ダウンロードまたは購入した Orbolt アセットを保存しておくためのパネルがあります。

デジタルアセットの作成

1 Houdini でノードとネットワークを作成します。

2 ネットワークをパッケージ化し、他のアーティストと共有可能な Houdini デジタルアセット (.hda) ファイルとして保存します。

3 パラメータとハンドルをアセットのトップレベルにプロモートすることで、アセットのインターフェースを構築します。



4 アセットを使用するには、.hda ファイルを Houdini に再度ロードします。

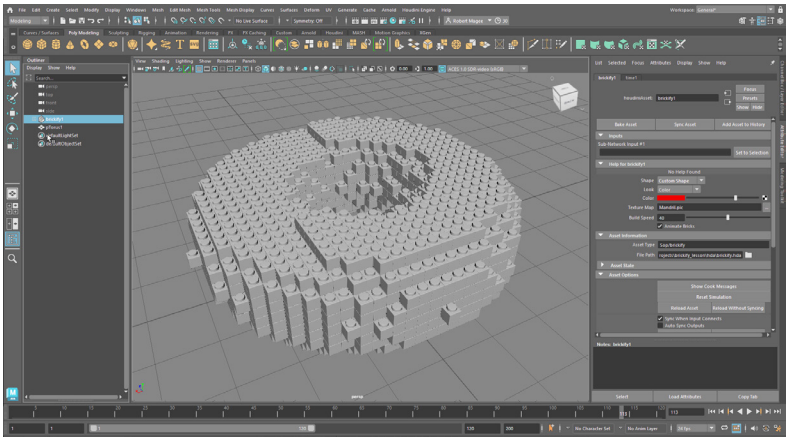
アセットレベルにプロモートされたパラメータのみ使用できます。他のすべてのパラメータはロックされます。

同じアセットを複数の Houdini シーンで何度でも使用できます。HDA ファイルに変更を加えると、他のすべてのアセットも簡単に同期されます。

HOUDINI ENGINE
他のアプリとの共有

Houdini Engine を使用すると、Houdini のプロシージャルなノードベースのアプローチを使い慣れたアプリケーションに取り込めます。このテクノロジを利用して Houdini デジタルアセットを共有すれば、同僚は Autodesk® Maya® や 3ds Max® などの 3D アプリケーションや、Unity® や Unreal® といったゲームエディタに直接アセットをロードできます。

他のアプリケーションを使用しているアーティストが Houdini デジタルアセットのメリットを享受できるのは、Houdini Engine プラグインのおかげです。Houdini Engine API で作成されたプラグインにより、.hda ファイルとすべてのハンドルおよびコントロールをホストアプリケーションにロードできます。アセットでパラメータが設定されると、Houdini は「内部」でノードとネットワークをクックし、その結果をホストに返します。



Houdini Engine を使用して Autodesk Maya にロードされた Houdini デジタルアセット

HOUDINI ENGINE API

これを実現しているのが Houdini API で、API を利用してホストアプリケーション向けのプラグインが作成されています。HAPI は、フラットかつ小規模な API で、習得も簡単です。独自のプラグインを作りたい開発者向けに、**Github** で提供されています。

HOUDINI ENGINE プラグイン

Houdini インストーラから、またはオンラインでアクセスできる Houdini Engine プラグインがいくつかあります。プロダクション環境で実証済みのこれらのプラグインなら、アーティストやスタジオは安心して利用できます。

各プラグインは、一般的な Houdini アセットの機能とホストアプリケーションの仕組みの間を橋渡しするように設計されています。例えば、ボリュームを使用するクラウドアセットは、Maya では問題なく機能するものの、ボリュームがサポートされていない Unity や Unreal では解釈されません。

無償の Houdini Engine for Unity/Unreal または**無償の Houdini Engine Indie** ライセンスで動作するプラグイン:

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

Houdini Engine ライセンスまたは**無償の Houdini Engine Indie** ライセンスで動作するプラグイン:

- **Autodesk Maya**
- **Autodesk 3ds Max**
- 独自開発のプラグイン

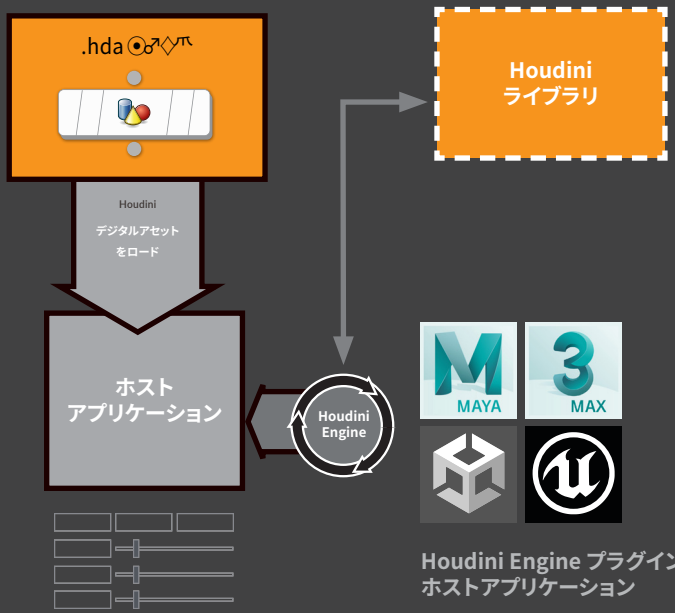
HOUDINI ENGINE パイプライン

1 Houdini Engine プラグインを使用して、.hda ファイルをホストアプリケーションにロードします。

2 ホストアプリケーションが、Houdini Engine を使用してアセットとインターフェースを受け取ります。

3 Houdini Engine が Houdini ライブラリファイルをコールし、アセット内のノードとネットワークを「クック」します。

4 アセットがロードされたり、パラメータが変更されると、Engine は Houdini ライブラリを取得し、ノードをクックしてから、結果をホストに返します。



Houdini Engine プラグインが動作するホストアプリケーション

映画および TV のパイプライン
アニメーションと VFX

実写プレートにビジュアルエフェクトを施すにしても、フル CG のショットを作成するにしても、映画や TV プロジェクトの最終目標は動画です。動画の映像は、キャラクタ、セット、エフェクトなどのアセットを最終的なコンポジットで 1 つにまとめあげることで、作られています。

Houdini は、映画および TV 番組制作パイプラインの全段階に使える、フル機能のパッケージです。モデリングからレンダリング、アニメーション、最終的なコンポジットまで、Houdini にはクリエイティブなプロセスを支えるプロシージャルなツールが搭載されています。長年にわたり、Houdini が業界標準になっているのは VFX の分野です。SideFX は、アカデミー賞でのオスカー獲得をはじめ、科学技術部門での賞をいくつも受賞しています。
プロシージャルモデリング、ライティング、キャラクタ作成といった分野も強化されており、スタジオからは熟練の Houdini アーティストを求める声が高まっています。

HOUDINI CORE / HOUDINI FX

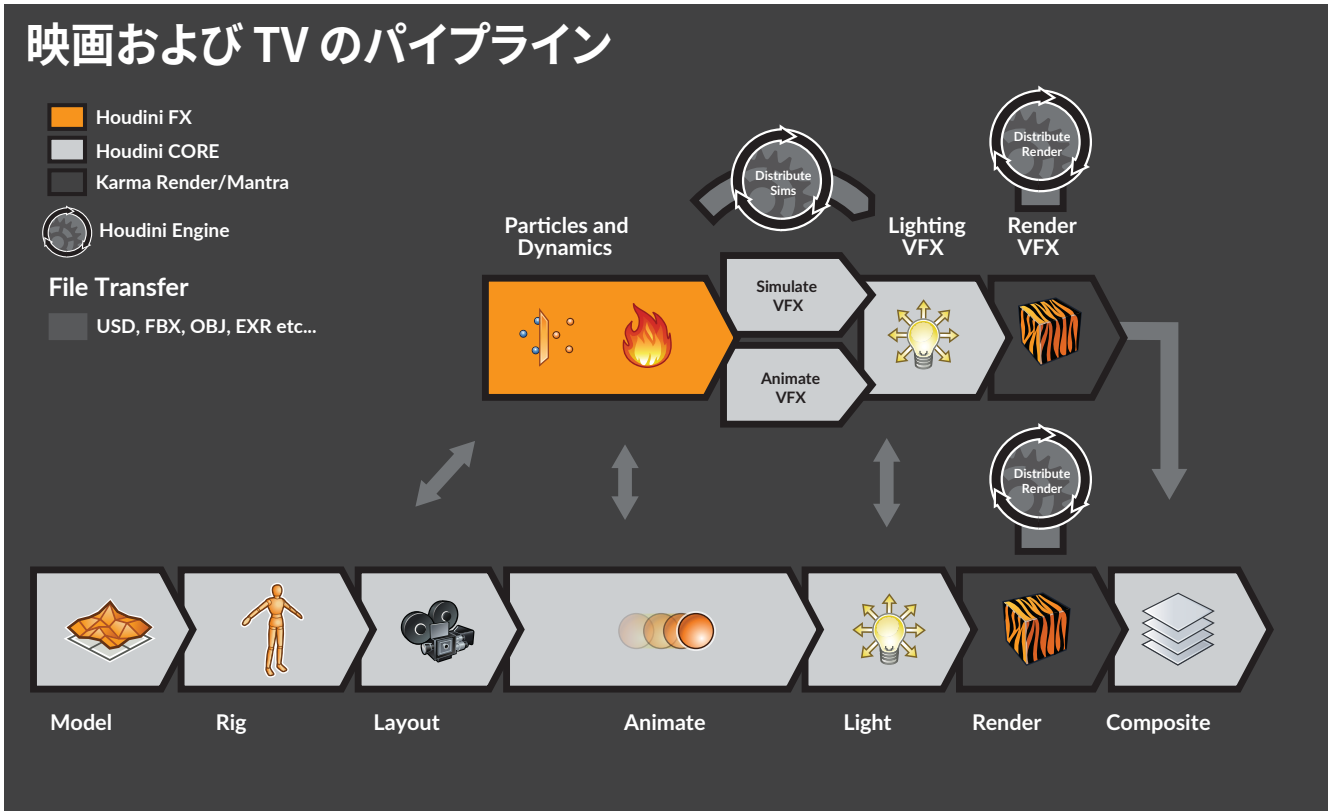
パイプラインで使用する Houdini には、2 つの商用バージョンがあります。Houdini Core は DOP を除くすべての Houdini ツールを搭載し、Houdini FX は完全なツールセットを搭載しています。Houdini FX で作成したシーンや VFX は、Houdini Core でステージング、アニメート、ライティング、レンダリングが可能です。FX アーティストは Houdini FX ライセンスで、それ以外のユーザは Houdini Core ライセンスを使用することで、堅牢なパイプラインが得られます。
例えば、シニアテクニカルディレクターが Houdini FX を使用して特定のプロダクションの課題を解決し、その結果のノードとネットワークを Houdini デジタルアセットにラップする使い方があります。アーティストフレンドリーな UI を構築し、アニメータや VFX アーティストがコスト効率のよい Houdini Core でショットを作るのを支援します。

相互運用性

ほとんどのスタジオは、多種多様な 3D アプリケーションを使用して、パイプラインの各領域に対処しています。Houdini には、異なるアプリケーション間でデータをやり取りできるようにする、相互運用性に優れたツールが多数あります。USD、Alembic、FBX、EXR のどれを使用していても、アーティストはさまざまな DCC アプリケーション間を簡単に行き来できます。また、Houdini Engine プラグインを使用すると、アセットのプロシージャルなコントロールを保持したまま、Autodesk® Maya® や 3ds Max® などのアプリケーションに Houdini デジタルアセットを取り込むことができます。
小規模なスタジオで、短納期に対応するプロダクションでは、大規模なファイル交換は避けたいことでしょう。フル機能のプロシージャルな既成パイプラインである Houdini なら、すべての段階を扱えます。

レンダリングとシミュレーションの分散

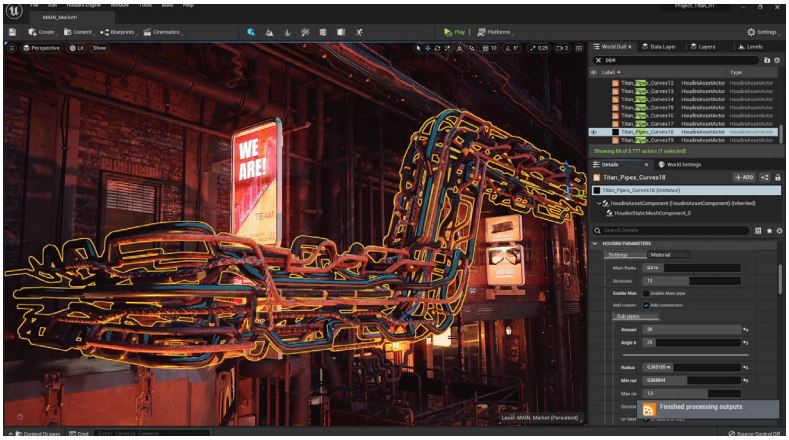
画像のレンダリングや VFX のシミュレーションは、時間がかかる処理です。フォトリアリスティックな結果を得たい場合には、さらに長い時間を要します。Houdini では、**Houdini Engine** の**バッチ**モードを使用して、レンダリングおよびシミュレーションタスクを演算ファームに分散させることができます。
シミュレーションの分散により、短時間で結果が得られるだけでなく、1 台のコンピュータではメモリ不足が懸念されるようなエフェクトも処理できます。シミュレーションをスライスして分散させることで、最終的な結果に妥協することなく、メモリを管理できます。**Houdini Engine** を使用して、ファームでシミュレーションを実行することを強くお勧めします。



ゲーム開発および VR のパイプライン インタラクティブ体験

ビデオゲームやバーチャルリアリティのプロジェクトで重視なのは、スムーズなゲームプレイ体験を実現するよう高度に最適化されたコンテンツを使用して、インタラクティブな 3D 世界を構築することです。映画に似たゲームシネマティクスのレンダリングとは、異なるタイプのパイプラインを用いることになります。

ゲームパイプラインの中核は、Unreal や Unity などのゲームエンジンです。これらのエンジンが、ゲームアートとゲームインタラクションを統合し、プレイ可能な体験を作り出します。ゲームアーティストは Houdini を使用して、地形の作成、レベルの設計と配置、プロシージャルモデルの構築、キャラクタの構築とアニメーション、火、流体、破壊などのリアルタイム FX の作成を行うことができます。



Houdini Engine を使用して Unreal にロードされた Houdini デジタルアセット

ゲームエンジンへのエクスポート

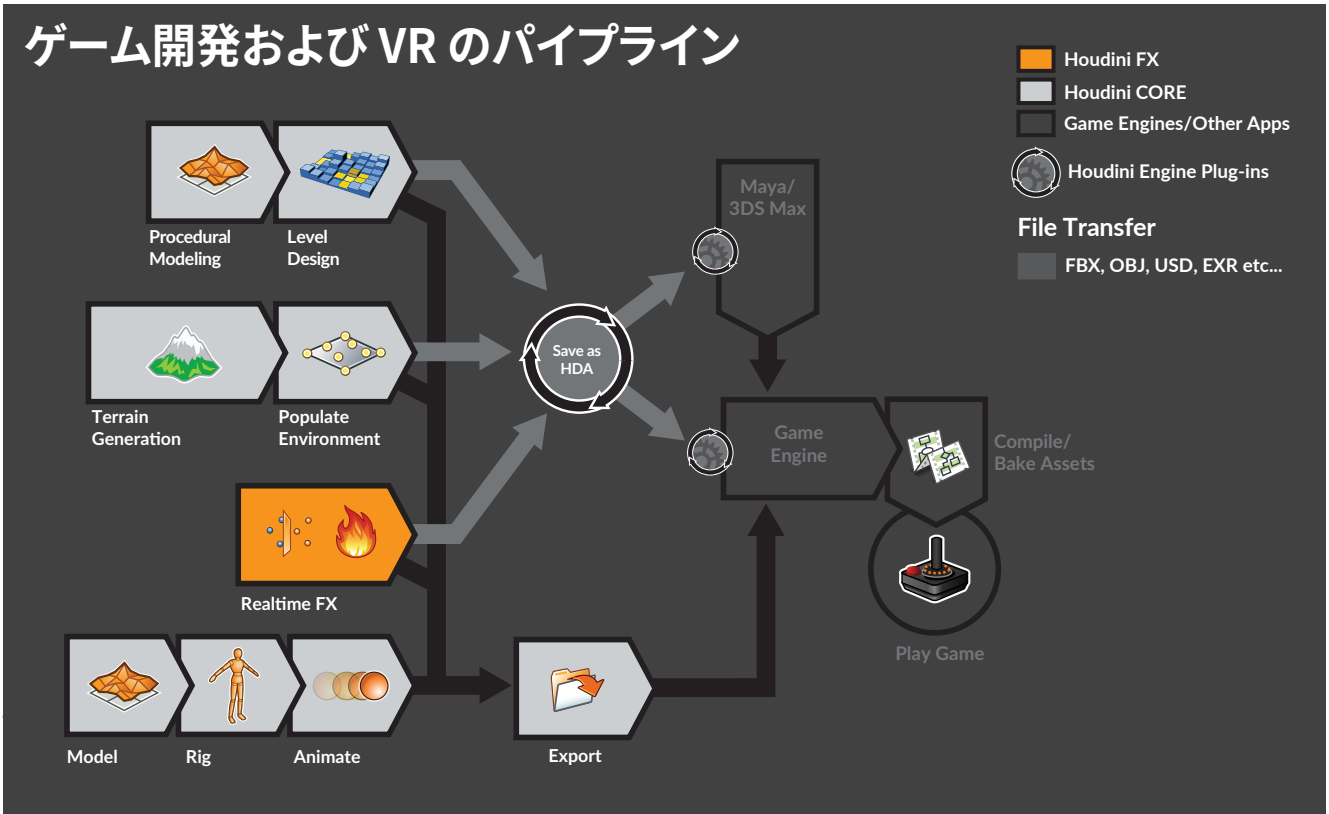
Houdini からゲームエンジンにコンテンツを取り込むには、2 通りの方法があります。従来は、FBX や OBJ といったファイルフォーマットにエクスポートし、それをエンジンにインポートしていました。Houdini でプロシージャルなシステムを作成し、その結果を平坦化します。

もう 1 つの方法は、**Houdini デジタルアセット**を作成し、Unreal や Unity 向けの **Houdini Engine プラグイン**を使用してゲームエンジンにロードすることです。これらのアセットは、パラメータやコントロールを保持したまま、ゲームエディタにインポートされます。ゲームエディタ内で変更を加えることができ、Houdini Engine がバックグラウンドで動作してアートワークを更新します。

ゲームアーティストはエディタ内でこのプロシージャ性を活用でき、ゲームをコンパイルする際に、アートワークがベイクされます。Houdini Engine はランタイムソリューションではないため、ゲームプレイの一部としてアクセスすることはできません。

リアルタイム FX

Houdini は VFX で知られており、ゲームの FX 作成に最適なツールです。しかし、FX は、テクスチャシート、フローマップ、頂点アニメーションテクスチャなどのテクニックを用いて最適化する必要があります。エフェクトのフットプリントを可能な限り軽くすることで、ゲームのフレームレートが保持され、コマ落ちも起きません。本ドキュメントで紹介した SideFX Labs ツールは、このようなワークフローをサポートするよう設計されています。



製品とライセンス

Houdini を使い始めるにあたり、Houdini 製品の種類を知っておくと役立ちます。大規模なスタジオ、小規模なスタジオ、新たに立ち上げた独立系（インディ）制作チームなど、それぞれのニーズに応じた各種 Houdini 製品があります。また、学校のラボや無償で学習したい学生のための Houdini もあります。

商用ライセンス

Houdini Core – モデラー、ライティングアーティスト、キャラクタリガー、アニメータ、ゲームアーティスト向けに設計され、コンポジティングやモーション編集などの機能も搭載しています。Houdini FX で作成したシーンは、Houdini Core で開いてレンダリングできるため、VFX のライティングに最適です。

Houdini FX – Houdini FX は、Houdini Core に含まれるすべてのツールに加えて、パーティクルおよびダイナミックシミュレーションツールも搭載しています。Houdini FX のプロシージャルワークフローを使用すると、流体、Pyro FX、粒、布、ヘアとファー、群衆、ソフトボディエフェクトなどを作成できます。

Houdini Engine – Houdini Engine は、コマンドラインからバッチモードで実行し、レンダリングや分散ダイナミックシミュレーションを一括で処理することができます。また、Autodesk® Maya®、Autodesk® 3ds Max®、Unity®、Unreal® をはじめ、他のデジタルコンテンツ作成アプリケーションに Houdini デジタルアセットをロードするときにも Houdini Engine が使用されます。

INDIE ライセンス

Houdini Indie - 起業したばかりのアニメータやゲーム制作者に、Houdini のアニメーションおよび VFX ツールのすべてを提供します。Houdini Indie の商用利用は、約 1000 万円以下の収益に制限されています。

Houdini Engine Indie – Houdini Engine Indie ライセンスを使用すると、Houdini Indie をバッチモードで実行したり、Houdini デジタルアセットを他のコンテンツ作成アプリにロードすることができます。

ライセンスについて

Houdini Education – Houdini Education は、学校、トレーニングセンター、学生による使用を目的として設計された、Houdini FX のフル機能バージョンです。Houdini Education では、Houdini Apprentice で作成したファイルを開くことができます。

Houdini Apprentice – Houdini Apprentice は、Houdini FX の無料体験バージョンで、学生、アーティスト、ホビーユーザが非商用目的の個人プロジェクトに使用できます。受賞歴のある Houdini FX のほぼすべての機能にアクセスできるので、スキルの強化や個人プロジェクトに役立ちます。Apprentice ではディスクにシーンを保存可能ですが、レンダリング画像にはウォーターマークが追加されます。

注意：Indie、Apprentice、Education で作成したシーンファイルとアセットは、商用バージョンの Houdini では使用できません。ファイルフォーマットが異なるうえ、異なるライセンスタイプ間でのファイル共有は EULA（エンドユーザー使用許諾契約書）で認められていません。

ライセンスタイプ

ワークステーション（ノードロック） – このライセンスタイプは 1 台のコンピュータ上で使用でき、ローカルサーバーまたは SideFX.com からのみアクセス可能です。

ローカル / グローバルアクセス（フローティング） – これらのライセンスはサーバー上にセットアップでき、アーティストチームで共有可能です。アーティストが Houdini を起動すると、利用可能なライセンスがある場合、ライセンスがサーバーからチェックアウトされます。ローカルライセンスは単一のスタジオ向けで、グローバルライセンスは異なる場所にあるスタジオで共有することを目的に設計されています。

ライセンスのインストール

ライセンスを取得したら、**Houdini License Administrator（hkey）**アプリケーションを開き、**File > Install Licenses** を選択して、ライセンスをインストールします。ログインとパスワードの入力を求められますが、これは SideFX.com の Web サイトで設定したものと同じです。ローカルにインストールするのではなく、sidefx.com をライセンスサーバーとして使用できるようになりました。**ログインライセンス**を使用するには、常に SideFX アカウントでログインしている必要があります。異なるコンピュータを使ってログインすることもできますが、一度に使用できるのは 1 台のみです。この方法は、独立系制作者や学生ユーザに最適です。

ローカルおよびグローバルアクセスライセンスは、この方法で中央サーバーにインストールできます。ライセンスにアクセスする全員が、そのサーバーを利用できるようにする必要があります。

また、ライセンスを確認したい場合は、SideFX.com の Web サイトの右上に表示されるアバターをクリックし、**Services** を選択して、**Manage Licenses** リンクをクリックします。

年間アップグレードプラン（AUP）

Houdini への投資を最大化する年間アップグレードプラン（AUP）は、ビジュアルエフェクトスタジオ、ゲームスタジオ、3D アーティストに重要な特典を提供します。プロダクションレベルのテクニカルサポート、最新のソフトウェア機能強化を含むフルリリースとドットリリース、バグ修正を含むデイリービルドなどをご利用いただけます。

SIDEFX サポート

Apprentice ユーザを含むすべてのお客様は、インストールやライセンスに関する問題について、メールサポートシステムで SideFX にお問い合わせいただけます。複雑なプロダクションに関する問題や質問は、年間アップグレードプランおよび商用レンタルのお客様のみ、サポートチームに問い合わせが可能です。

弊社のサポート担当窓口は、**support@sidefx.com** です。

次の情報をメールに含めてください。

- ***お使いのオペレーティングシステム（Windows XP など）***
- ***Houdini のバージョンおよびビルド番号***
- ***インストールの問題の概要、ライセンスの問題がある場合には診断ファイル***

サポートプロラムの詳細については、**SideFX.com/support** をご覧ください。

比較表

		商用		独立系(インディ)	学習	
製品		HOUDINI FX	HOUDINI CORE	HOUDINI INDIE	EDUCATION	APPRENTICE
対象ユーザ		スタジオ	商業アーティスト	独立系	フリーランス	学校	学生	ホビーユーザ
価格		SideFX.com を参照		1 年間 $269 USD	1 年間 $75 USD	無償
オペレーティングシステム		Windows、LINUX、Mac OSX				
機能	モデリング	✓	✓	✓	✓	✓
	キャラクタ	✓	✓	✓	✓	✓
	アニメーション	✓	✓	✓	✓	✓
	Solaris : レイアウトツール	✓	✓	✓	✓	✓
	Solaris : ルックデブとライティング	✓	✓	✓	✓	✓
	Karma/Mantra レンダリング	✓	✓	✓	✓	✓
	地形	✓	✓	✓	✓	✓
	合成	✓	✓	✓	✓	✓
	ボリューム	✓	✓	✓	✓	✓
	Pyro FX	✓	Simple Fireball	✓	✓	✓
	流体	✓	シンプルな Flip	✓	✓	✓
	リジッドボディ	✓	Simple Fracture	✓	✓	✓
	パーティクル	✓	-	✓	✓	✓
	Vellum Cloth	✓	Simple Cloth	✓	✓	✓
	ワイヤーダイナミクス	✓	-	✓	✓	✓
	群衆	✓	-	✓	✓	✓
ライセンス		商用		限定的な商用	非商用	
	ワークステーション(ノードロック)	✓	✓	✓	-	✓
	ローカル/グローバルアクセス(フローティング)	✓	✓	-	✓	-
ユーザインターフェース						
	Houdini GUI アクセス	✓	✓	✓	✓	✓
	コマンドラインアクセス	✓	✓	✓	✓	✓
	GUI ウォーターマーク	-	-	透かし入り(小)	透かし入り(小)	透かし入り(小)
	プラグインサポート	✓	✓	✓	✓	✓
HOUDINI ENGINE						
	HOUDINI ENGINE プラグイン	✓	✓	✓	✓	なし
	Engine 用アセット作成	✓	✓	✓	✓	Education ライセンス用
	Orbolt 用アセット作成	✓	✓	✓	✓	✓
レンダリング						
	Karma トークン	5 / 10*	5 / 10*	1	10	1
	Mantra トークン	無制限	無制限	1	10	1
	サードパーティ製レンダラ	✓	✓	✓	✓	なし
	ウォーターマーク付きレンダリング	-	-	-	-	✓
	解像度	無制限	無制限	無制限	無制限	1280 x 720
フォーマット	シーン					
	.hip	✓	✓	.hipalc	.hipanc	.hipanc
	.hda	✓	✓	.hdalc	.hdanc	.hdanc
	ジオメトリ					
	USD	✓	✓	✓	✓	.usdnc
	FBX	✓	✓	✓	✓	インポート
	Alembic	✓	✓	✓	✓	インポート
	.bgeo	✓	✓	✓	✓	✓
	画像					
	.pic	✓	✓	.piclc	✓	.picnc
	.exr	✓	✓	✓	✓	ウォーターマーク付き
	.tif	✓	✓	✓	✓	ウォーターマーク付き
	.png/.jpg	✓	✓	✓	✓	ウォーターマーク付き

* 商用のワークステーションライセンスには Karma トークンが 5 つ付属し、ローカルおよびグローバルアクセスライセンスには Karma トークンが 10 個付属しています。

HOUDINI FOUNDATIONS

モデリング、レンダリング、アニメーション

Houdini へようこそ。このレッスンは、サッカーボールのモデリング、レンダリング、アニメーション、シミュレーションをゼロから行います。スクワッシュ&ストレッチ（潰しと伸ばし）の理論を使用して昔ながらのバウンシングボールのアニメーションを作成し、テクスチャとマテリアルを適用し、ライトとカメラを追加します。また、ダイナミクスを使用してサッカーボールの一群をシミュレートする方法も探ります。

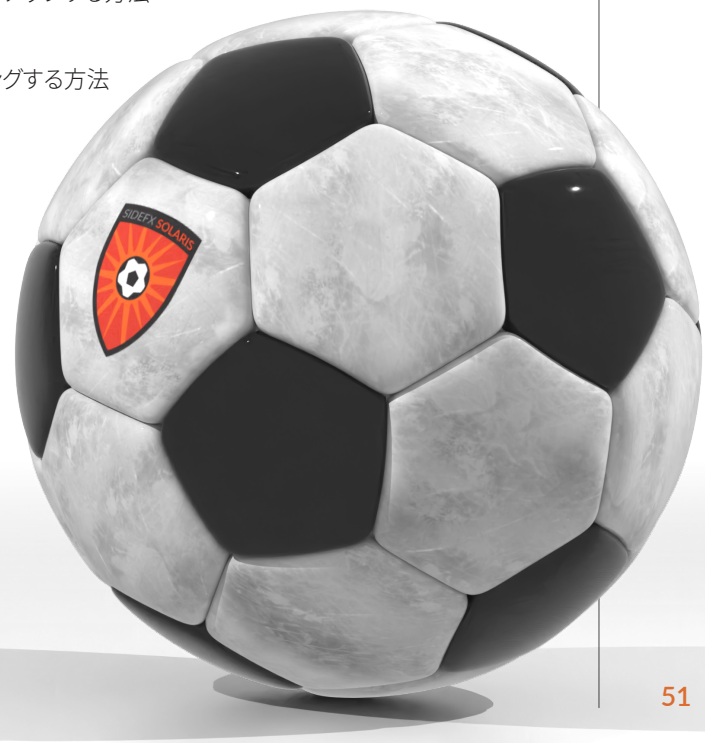
はじめての Houdini シーンを作成し、Houdini のさまざまな部分を確認し、インターフェースや必須のツールなどを実際に使っていきます。**Scene View** でインタラクティブに作業する方法と、**ネットワークビュー**を使用してノードを管理しながら、モデルを整えたり、アニメーションリグを構築する方法を学びます。また、**Solaris ステージ**でマテリアルとテクスチャをセットアップしてから、Houdini の組み込みのレンダラである **Karma** を使ってレンダリングし、最後には**リジッドボディシミュレーション**を作成します。

レッスンの目標

Houdini のプロシージャルなノードベースのワークフローを使用して、サッカーボールのモデリング、レンダリング、アニメーション、シミュレーションを行います。

学習内容

- **View ツール**を使用する方法
- シェルフ、**Radial メニュー**、**Tab** キーを使用する方法
- **ジオメトリ**を作成する方法
- **ノードとネットワーク**を使用する方法
- **カスタムアトリビュート**と **For-Each ループ** をセットアップする方法
- **マテリアル**と**テクスチャ UV** をセットアップする方法
- ショットの**レイアウト**を作成し、**Karma** でレンダリングする方法
- **キーフレームを設定**し、**Motion FX** を追加する方法
- **リジッドボディダイナミクス**を使用する方法



使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、以下の Houdini 製品で実行可能です。

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

ドキュメントバージョン 4.0.1J | 2023 年 8 月

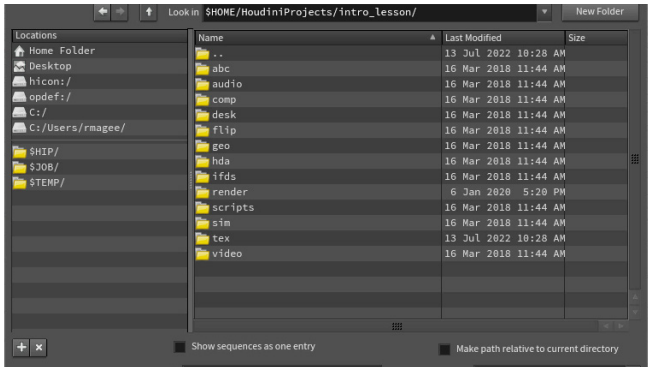
パート 1

Houdini UI の確認

はじめに、Houdini ワークスペースと、頻繁に使うことになる 3 つのペインを理解しておきましょう。**ビューポート**ではオブジェクトをインタラクティブに作成でき、**パラメータエディタ**ではノードプロパティを編集でき、**ネットワークエディタ**ではノードネットワークを直接操作することができます。

プロジェクトファイル

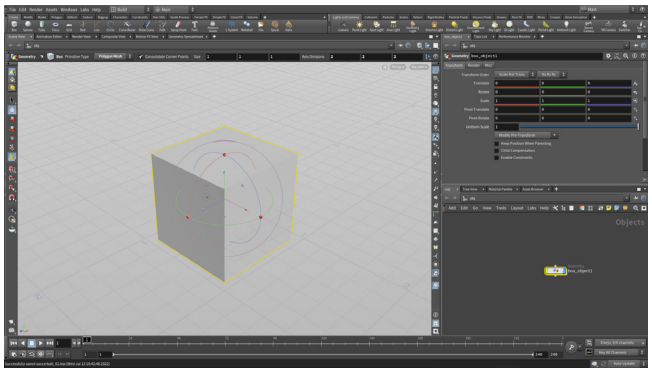
SideFX.com の **サッカーボールチュートリアル**のページ（このドキュメントを入手した場所）から、***intro_lesson*** ディレクトリをダウンロードします。**home** または **documents** ディレクトリにある、**Houdini Projects** ディレクトリに配置してください。



01

File > Set Project を選択します。ダウンロードした ***intro_lesson*** ディレクトリを見つけ、**Accept** を押します。これで、このショットに関連するファイルがすべて、先ほどコピーしたプロジェクトディレクトリとそのサブフォルダに配置されるようになります。

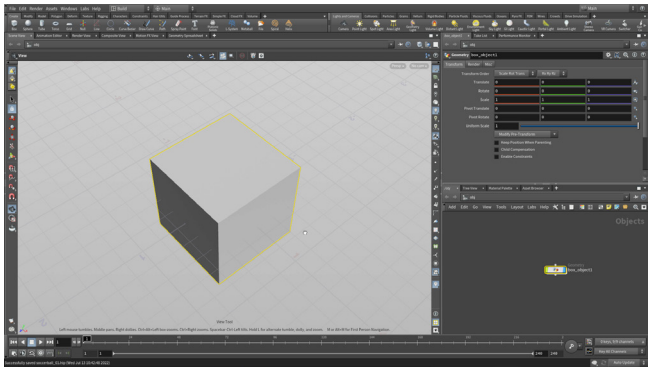
File > Save As... を選択すると、新しい ***intro_lesson*** ディレクトリが表示されます。ファイル名を ***soccerball_01.hip***（または ***football_01.hip***）に設定し、**Accept** をクリックして保存します。



02

ビューポートで、**C を押して** Radial メニューを表示します。このメニューから **Create > Geometry > Box** を選択します。カーソルの位置に、シーン内への配置待ちの状態にあるボックスの輪郭が表示されます。**Enter を押して**、原点の位置に配置します。

Scene View にボックスが作成され、ネットワークエディタにノードが追加され、パラメータエディタにはオブジェクトパラメータが表示されます。これらの各種インターフェース要素を使用しながら、プロジェクトを進めていきます。



03

Houdini の **View** ツールを見ていきましょう。次のホットキーを押してください。

- **タンブル** スペースバーまたは Alt（Opt）+ LMB クリック + ドラッグ
- **パン** スペースバーまたは Alt（Opt）+ MMB クリック + ドラッグ
- **ドリー** スペースバーまたは Alt（Opt）+ RMB クリック + ドラッグ

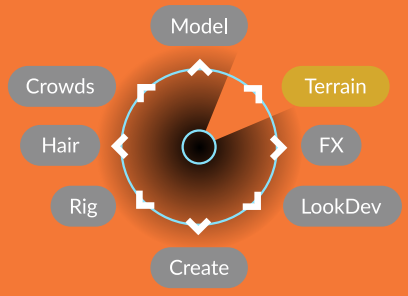
ホームをすることで、ビューをリセットしたい場合もあります。そうした場合に便利なホットキーもあります。

- **Home Grid** スペースバー + H
- **Home All** スペースバー + A
- **Home Selected** スペースバー + G

Radial メニュー

Houdini のツールには、**X**、**C**、**V** のホットキーを使ってアクセスできる Radial メニューからもアクセスできます。いずれかのホットキーを押すと、Radial メニューが表示され、各種オプションを選択できます。各メニューの主な内容は次の通りです。

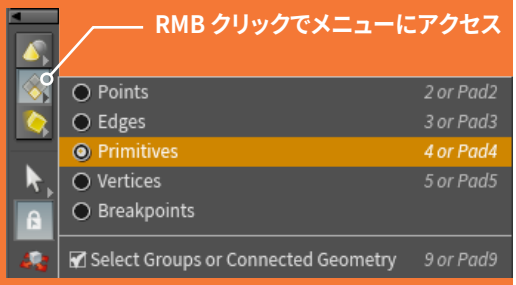
スナップ	**X**
メイン（またはカスタム）	**C**
ビュー	**V**

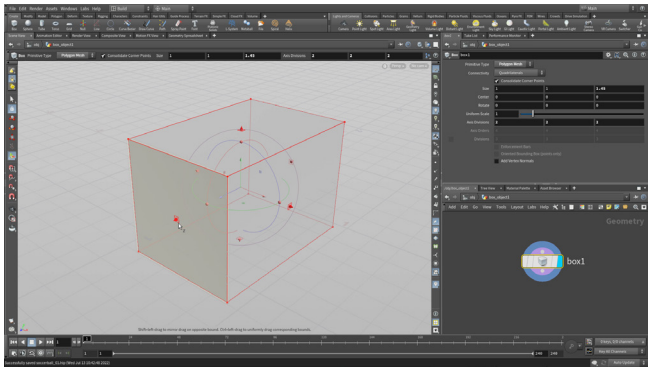


選択のホットキー

Select、**Move**、**Rotate**、**Scale**、**Handles** ツールを使用する場合、次のホットキーで選択モードと作業するレベルを決定します。

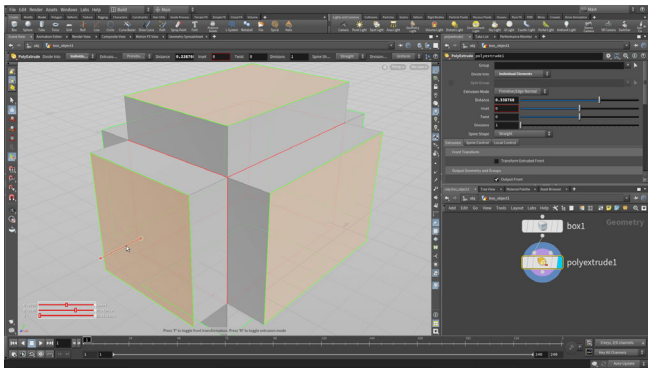
オブジェクト	**オブジェクトレベル**	**1**
ポイント	**ジオメトリレベル**	**2**
エッジ	**ジオメトリレベル**	**3**
プリミティブ / 面	**ジオメトリレベル**	**4**
頂点	**ジオメトリレベル**	**5**





04 オブジェクトを選択した状態で、**I を押し**、そのジオメトリレベルに移動します。**Shift** キーを使ってハンドルをドラッグし、原点を中心に Z 軸に沿って長くします。

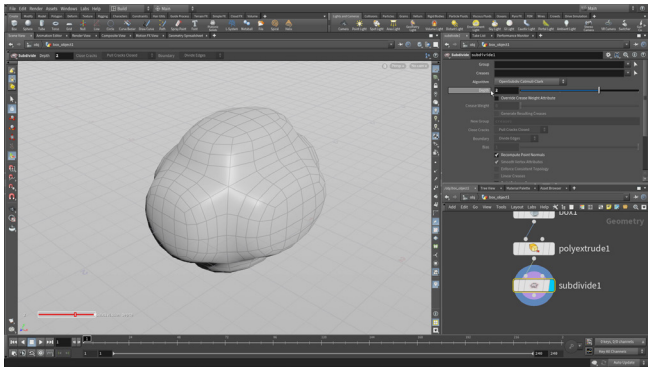
Houdini でオブジェクトを作成すると、オブジェクトのトランスフォームを管理する**オブジェクトレベル**と、形状を定義する**ジオメトリレベル**ができます。I を押すと、このオブジェクトのジオメトリレベルに入ることができます。また、ネットワークエディタでオブジェクトノードをダブルクリックしても、ジオメトリレベルに入れます。後でオブジェクトレベルに戻るには、**U を押します**。



05 **S を押して Select ツール**に切り替えたら、**4** を押してプリミティブの選択にアクセスします。**N を押して**すべてを選択してから、**C を押して** Radial メニューを表示し、**Model > Polygons > Poly Extrude** を選択します。

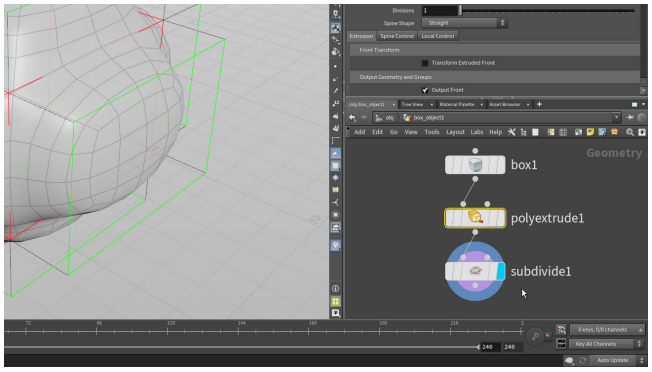
Scene View の上部にある**オペレーションコントロールツールバー**または**パラメータ**エディタで、**Divide Into** を **Individual Elements** に設定し、ハンドルを使用して **Distance** を約 **0.4** に設定します。これにより、各プリミティブの法線に沿って面が押し出されます。

ネットワークビューには 2 つのノードが表示されています。Houdini では手順を進めるたびにノードが作成され、それらのノードでシーンを調整していきます。



06 **N** を押して新しい面をすべて選択したら、**Tab** を押して **sub...** と入力し、リストから **Subdivide** を選択します。Houdini のツールにアクセスするには、**Tab** キーも便利な方法です。ツール名を入力すると、リストに候補のツールが表示されるので、サブメニューをナビゲートしなくても必要なツールを簡単に見つけることができます。

パラメータエディタで、**Depth** を **2** に設定します。ジオメトリが細分化され、ポリゴンの数が増えます。Houdini には、オブジェクトレベルで細分化を表示できるオプションもあり、ジオメトリを実際に追加しなくても細分化を確認できます。しかしここでやりたいことは、ポリゴンの作成です。



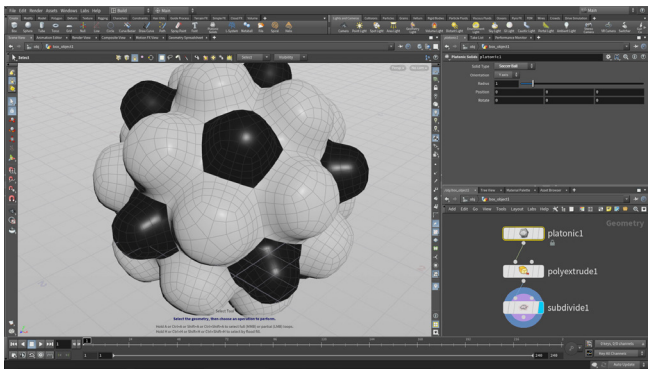
07 チェーン内のノードをいくつか選択してみましょう。選択したノードのハンドルが表示されますが、表示は最終的な形状のままです。各ノードの **Display フラグ**を設定し、表示ノードを変更してみましょう。**バイパス**や**テンプレート**といったフラグも試してください。***polyextrude*** ノードをネットワークから**移動して**、その後同じ場所に戻します。

最後に、すべてを元に戻し、***subdivide*** ノードの **Display フラグ**を設定します。これは非常に重要です。Display フラグによって、オブジェクトレベルでの表示が決まるからです。**Display フラグは必ず正しく設定してください！**

パート 2

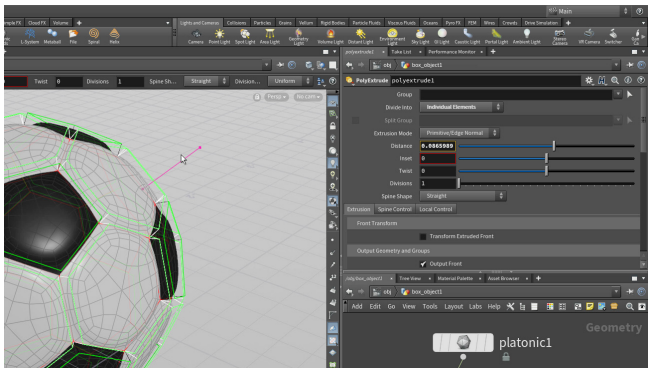
サッカーボールの作成

次は、ボックスをサッカーボールの形をしたプラトン立体に置き換えます。Houdini のプロシージャルなアプローチなら、Box ノードを Platonic Solids ノードに置き換えられます。そこから他のノードを調整して、サッカーボールらしい見た目にします。入力ノードの置き換えができるため、シンプルなジオメトリでネットワークのプロトタイプを作成しておけば、柔軟性が高まります。



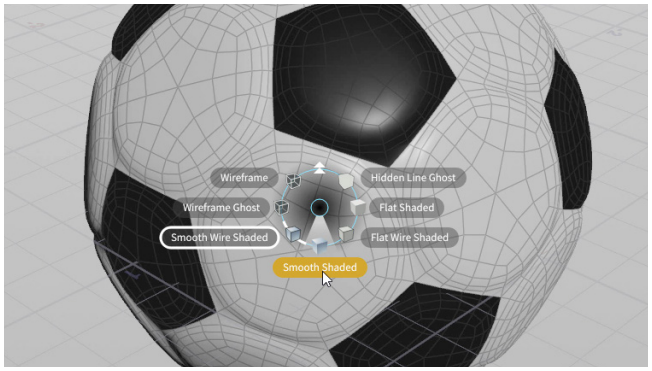
01 **ネットワークエディタ**で、**Tab** キーを使用して **Platonic Solids** ノードをネットワークに追加します。クリックしてチェーンの最上部付近に配置します。***platonic*** ノードを ***polyextrude*** に接続します。パラメータエディタで、**Solid Type** を **Soccer Ball** に設定します。***box*** ノードを選択して削除します。

Houdini はプロシージャルのため、入力ノードを変えても、ネットワーク全体が適切に機能することは珍しくありません。作業の柔軟性が高まるだけでなく、変更後の結果が気に入らなければいつでも、元の形状を接続し直せばよいわけです。



02 ***polyextrude*** ノードを選択します。**Handle** ツールがアクティブになっていることを確認したら、ビューポートでハンドルを使用して、**Distance** の値を下げます。パラメータエディタでパラメータ値を設定してもかまいません。これでサッカーボールらしい見た目になりました。***subdivide*** ノードを表示している場合でも、***polyextrude*** ノードを選択すると、そのハンドルとパラメータにアクセスできることを覚えておきましょう。

このプリミティブタイプで準備がすべて整ったと思うかもしれませんが、実際には、平らな面で構成された切頂正二十面体にすぎません。丸いサッカーボールが必要なので、もう少し手を加える必要があります。



03 ビューポートで **V** を押し、**Radial メニュー**で **Shading > Smooth Shaded** を選択します。ビューポートの右上のメニューを使用しても、シェーディングを変更できます。

このサッカーボールは、正式な革製のサッカーボールではなく、安いビニールボールのように見えます。これからノードを追加し、分岐させながら見た目を向上させていきます。

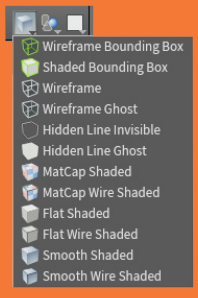
分析を終えたら、シェーディングの設定を **Smooth Wire Shaded** に戻します。

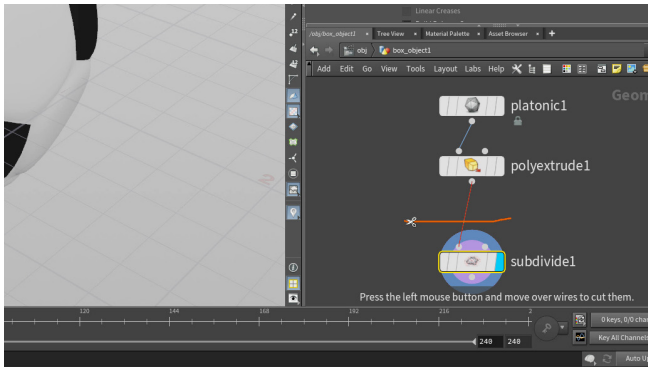
シェーディングオプション

View Radial メニューまたはビューポートの右上にある **Shading** メニューから、さまざまな**シェーディングオプション**を使用できます。

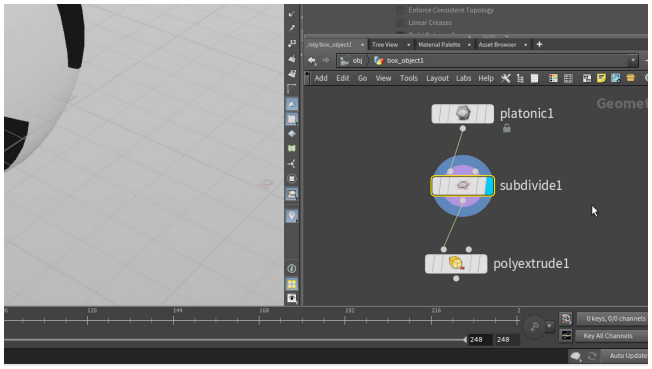
オブジェクトのシェーディングでは、ビューポートの右端にある**表示オプション**でライティングが決まります。ヘッドライト、通常のライティング、シャドウ付きの高品質なライティングのいずれかを選択できます。

シェーディングビューからワイヤーフレームに素早く切り替えるには、**W** キーを押します。

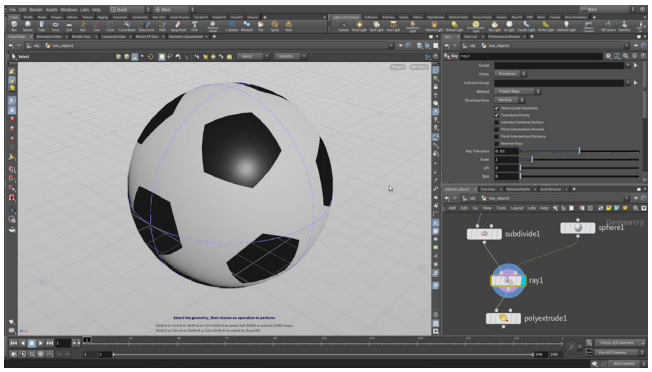




04 ネットワークビューで **Y を押し**、***subdivide*** ノードと ***polyextrude*** ノードをつなぐワイヤーを横切るようにドラッグして、接続を解除します。より丸みのあるサッカーボールになるように、subdivide を残り 2 つのノードの間に移動します。

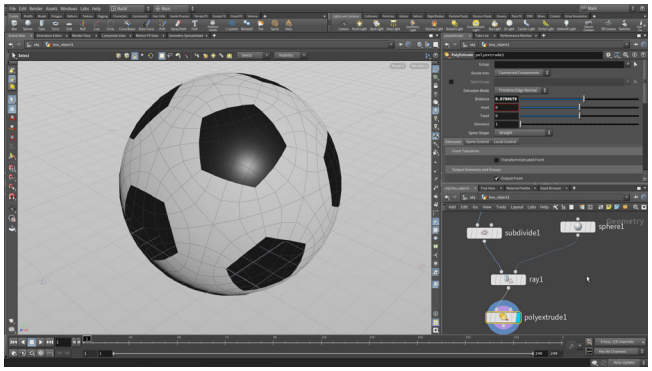


05 ***platonic*** ノードと ***polyextrude*** ノードの間に、***subdivide*** ノードを**ドラッグ**します。接続されたワイヤー上にドロップすると、自動的に挿入されます。挿入されない場合は、接続されるまで少し動かしてみてください。押し出される前に、球のディテールが増加します。



06 ネットワークエディタで **Tab キー**を使用して **Ray** ノードを追加し、***subdivide*** の後に接続します。次に、**sphere** ノードをネットワークに追加して、**Radius** を **1, 1, 1** に、**Primitive Type** を **Primitive** に設定します。sphere を ***ray*** ノードの 2 つ目の入力に接続します。これにより、細分化されたボールが完全な球に投影されます。

これは、Houdini の非常に強力なノードです。あるジオメトリのポイントを別のジオメトリに投影します。細分化されたサッカーボールが完全には丸くないという、ここでの問題にぴったりの解決策です。



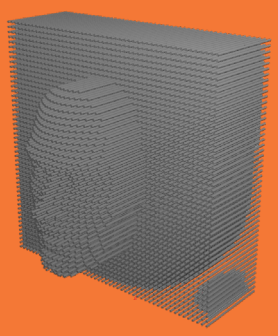
07 **polyextrude ノードに Display フラグを再度設定します。Divide Into** を **Individual Elements** に設定すると、すべての小さいポゴンが押し出されますが、それは望ましくありません。**Connected Components** に設定して、すべてのポリゴンが押し出されるようにします。

このネットワークでは、ボールが細分化された後で、サッカーボールの元のパッチを押し出す方法が必要です。これを行うには、元のジオメトリのプリミティブ番号を使用します。

Ray ノード

Ray ノードは、ポイントを別のジオメトリに投影するツールです。おもちゃのピンアートに似ています（子供の頃に遊んだことはありませんか）。実際、これは Houdini でピンアートのボードをセットアップするために使用するノードです。

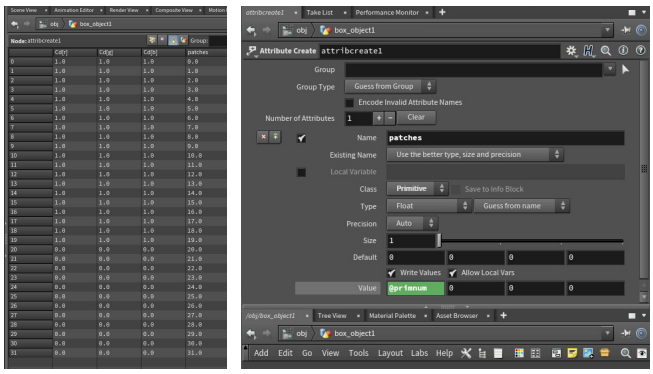
ヘルプの確認 | 各ノードについて詳しく知りたい場合は、パラメータエディタの右上にある ? ヘルプボタンをクリックすると、そのノードのオンラインドキュメントが表示されます。シェルフのツール上にカーソルを置いて、**F1 を押して**もヘルプを確認できます。Houdini で開けるサンプルファイルが用意され、そのノードの機能を学べるようになっているヘルプもたくさんあります。



パート 3

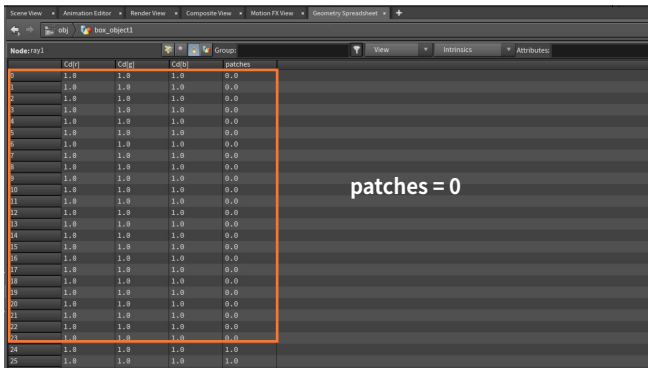
For Each ノード

ここで魔法が働きます。最後に作成したアトリビュートを For-Each ループに送り、それぞれの元のパッチに多数のポリゴンが含まれていても、それらのパッチが押し出されるようにします。ポリゴンを押し出した後、もう一度細分化すると、サッカーボールが革製らしく見えるようになります。



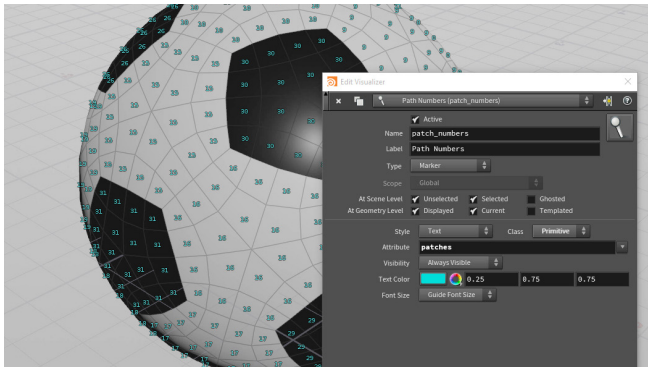
01 **AttributeCreate** ノードを ***platonic*** ノードの後に追加します。**Name** を ***patches*** に、**Class** を **Primitive** に設定します。**Value** フィールドの最初の値を **@primnum** に設定します。

このエクスプレッションは、プリミティブ番号アトリビュートを取得し、それを ***patches*** という新しいアトリビュートに変換します。



02 ***attributecreate*** ノードを選択した状態で、メインのビューポートの横にある **Geometry Spreadsheet** タブをクリックします。**Primitive** ボタンをクリックすると、左側にプリミティブ番号が表示され、パッチの色を示す 3 つのカラーアトリビュートと、プリミティブ番号と一致する ***patches*** アトリビュートも表示されます。

ray ノードをクリックします。このアトリビュートは、形状が細分化されるときに繰り越されます。もっと多くのプリミティブがあることが分かりますが、***patches*** アトリビュートは 31 までで、その後は 0 に戻ります。

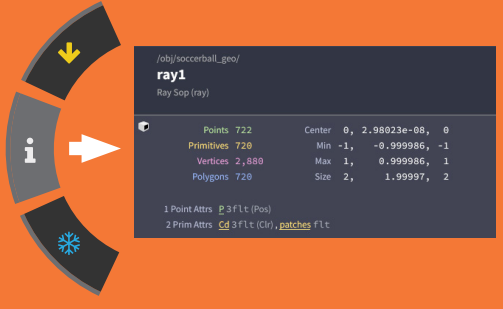


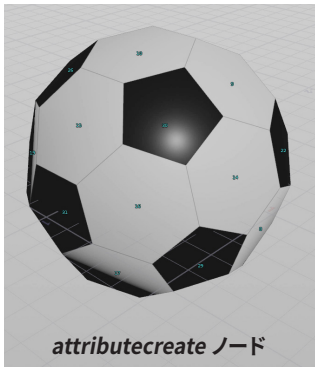
03 **Scene View** タブに戻り、**Display Options** バーの **Visualization** ボタンを **RMB クリック**し、**Scene** の横にある **+（プラス）記号**をクリックして、**Marker** を選択します。Edit Visualizer パネルで、**Name** と **Label** を ***Patch_Numbers*** に、**Type** を **Marker** に、**Class** を **Primitive** に、そして **Attribute** を ***patches*** に設定します。

アトリビュートの使用

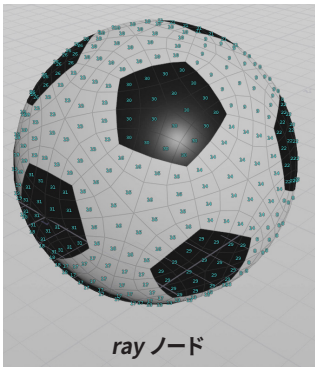
アトリビュートは、ポイント、プリミティブ、頂点に割り当てることができます。一般的なアトリビュートには、**カラー（Cd）**や **UV** などがあります。ノードにカーソルを置いて Radial メニューから **i** を選択すると、チェーン内の任意の場所でアトリビュートを確認できます。また、**Geometry Spreadsheet** パネルでもアトリビュート値を確認することができます。

このレッスンでは、***patches*** というカスタムアトリビュートを作成し、For-Each ループで活躍してもらいます。





attributecreate ノード

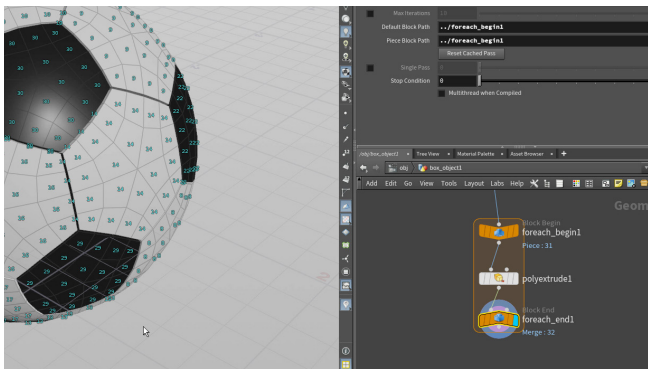


ray ノード

04

Visualization ボタンがオンになっていることを確認します。ビューポートのサッカーボール上にパッチの値が表示されています。元のプラトン立体からのプリミティブ番号が、細分化された面に転送されています。**Display フラグ**を別のノードに設定して、関係を確認してみましょう。この情報は、For-Each ループを使用して適切にパッチを押し出すために使用されます。

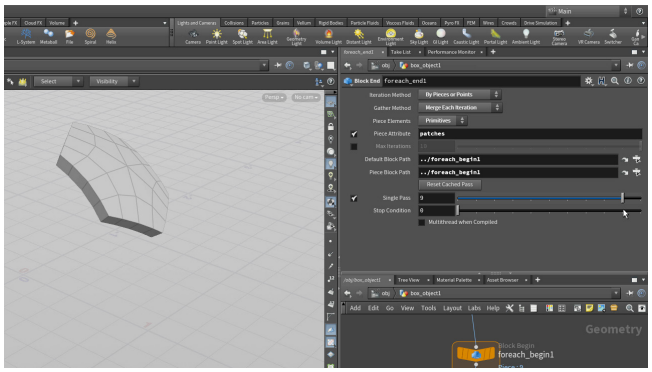
Visualization ボタンを**オフ**にして、作業内容を**保存**します。ここまでの手順について、やや抽象的で専門的すぎると感じているかもしれませんが、大丈夫です。努力はもうすぐ実を結びます。



05

ネットワークエディタで **Tab** を押し、**For-each Named Primitive** と入力して、2 つのノードにアクセスしてシーンに配置します。***ray*** ノードと ***polyextrude*** ノードの間に ***foreach_begin*** を接続し、***polyextrude*** ノードの後に ***foreach_end*** を接続します。***foreach_end*** ノードを選択し、パラメータエディタで **Piece Elements** は **Primitives** に設定したままにし、**Piece Attribute** を ***patches*** に設定します。***foreach_end*** ノードに Display フラグを設定します。

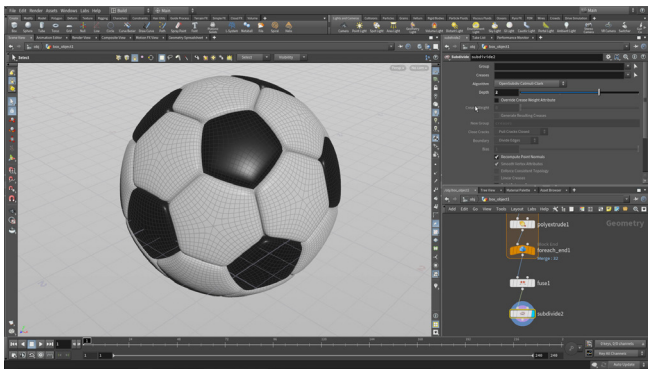
patches アトリビュートに基づいて、元のパッチが一斉に押し出されているのを確認できるはずです。押し出されていない場合は、***polyextrude*** ノードで **Divide to** が **Connected Components** に設定されていることを確認してください。



06

Single Pass の横にあるチェックボックスをクリックし、どうなるかを確認します。スライダをドラッグして、それぞれのパッチが個別に押し出される様子を確認してください。10 よりも大きい値に設定すると、さらに多くのパッチを確認できます。

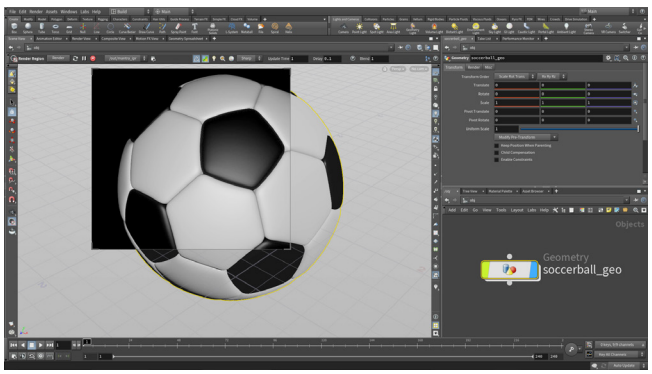
Single Pass をオフにして、完全な形状を表示します。***for-each*** ノードがすべてのパッチを作成し、最終的なジオメトリを返します。For-Each ループは、Houdini で頻繁に使用する強力なノードセットです。



07

foreach_end ノードの後に **Fuse** ノードを追加し、その **Display フラグ**を設定します。これは、ピースをつなげて単一のトポロジにするノードです。***for-each*** ノードでは、バラバラのパッチに分割しましたが、それらを再結合することはしませんでした。

Fuse の後に **Subdivide** ノードを追加します。**depth** を **2** に設定します。ビューポートでディテールが加わり、これを使用してモデルを評価できるようになります。ポリゴンの数は増えますが、まだ真の細分化サーフェスとしてレンダリングされるわけではありません。Houdini では、ジオメトリを追加することなく、ビューポートで細分化表示を設定することも可能ですが、この Subdivide ノードはレッスンの後半で必要となります。



08

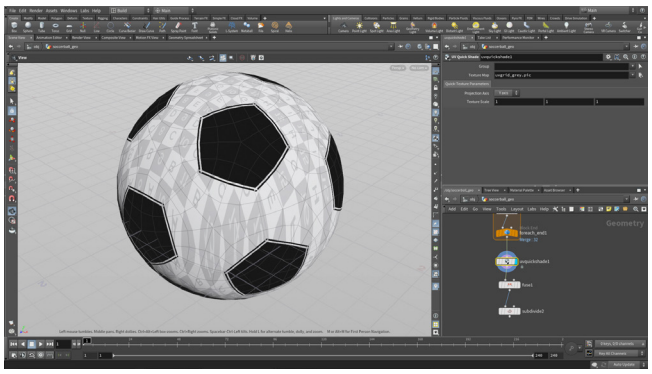
サッカーボールが革製に見えるように、***polyextrude*** の値を変えて試しましょう。ここでは **Distance** を **0.1**、Inset を **-0.02** に設定してください。パッチが丸みを帯びて、ルックが向上します。

オブジェクトレベルに移動し、パラメータエディタでオブジェクトの名前を ***soccerball_geo*** に変更します。ノードを選択するか、**F2** を押すか、名前を**ダブルクリック**します。**Render** タブをクリックし、**Render Polygons as Subdivisions (Mantra)** をオンにして、レンダリング時の真の細分化をセットアップします。**Render Region** ツールを選択したら、ビューポートでサッカーボールの周りにボックスを描き、プレビューレンダリングを作成します。キャンセルするには、その領域の右上にある **X ボタン**をクリックします。

パート 4

UV のセットアップ

マテリアルやテクスチャをセットアップするには、オブジェクトに適切な UV がセットアップされていることが大切です。Houdini のジオメトリには UV が付属していないため、自分で作成しなければなりません。つまり、ネットワークにノードを追加する必要があり、ここでは UV Quickshade と UV Flatten ノードを追加します。

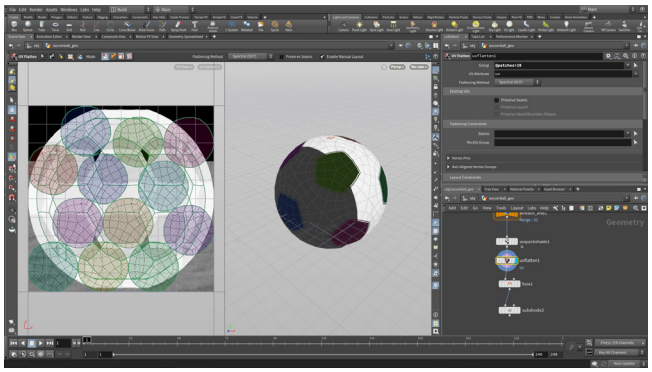


01

soccerball_geo オブジェクトの中に入ります。ネットワークビューで、Tab > **UV Quickshade** を押し、***foreach_end*** ノードのすぐ後に新しいノードを追加します。**Display フラグ** を設定します。

Texture Map の横にある**ファイル選択**ボタンをクリックします。`/tex` フォルダに移動し、***soccerball_color.rat*** を選択します。このテクスチャマップがジオメトリに表示されますが、投影方法を使用して UV を作成したため、全体が引き伸ばされたように見えます。

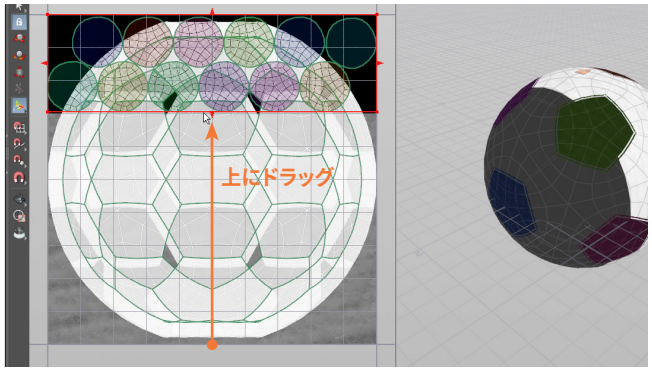
注：既存のテクスチャに合わせて UV をセットアップするのは、通常の作業順ではありません。逆の順番をあえて使っているのは、自分でテクスチャをペイントする必要がないからです。



02

Scene View で、**N を押して**すべてを選択してから、**Tab > UV Flatten** を選択します。**Group** フィールドで、`@patches>19` というエクスプレッションを入力します。これにより、パッチ境界を使用してサッカーボールジオメトリの暗いパッチが平坦化され、UV がレイアウトされます。

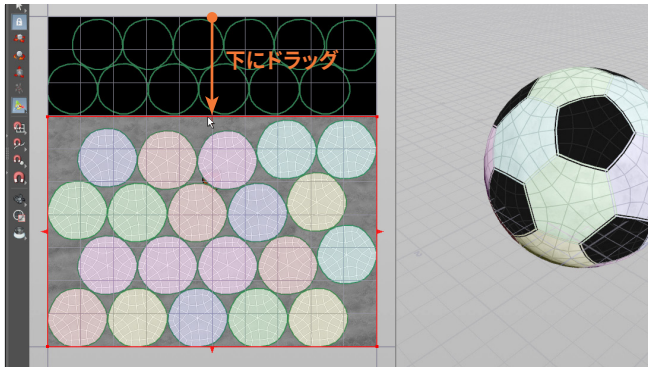
このツールを使用すると、UV ビューが表示されます。右上の **UV (vertex)** メニューをクリックして、**Background > *soccerball_color.rat*** を選択します。すると UV パネルの背景にこのテクスチャが表示されます。このテクチャには、画像の中央にチームのロゴが、上部には暗いパッチのための暗い領域が含まれています。



03

Handle ツール を **RMB クリック**し、***Min:*** ハンドルをオンにします。下部にある矢印ハンドルを、すべてのパッチがテクスチャマップ上部の暗い領域内に入るまで引き上げます。

このノードを ***fuse*** または ***subdivide*** の後に配置していたら、使用する境界はなかったでしょう。Houdini では既存のネットワークの途中で UV をセットアップでき、作業に柔軟性を持たせられます。



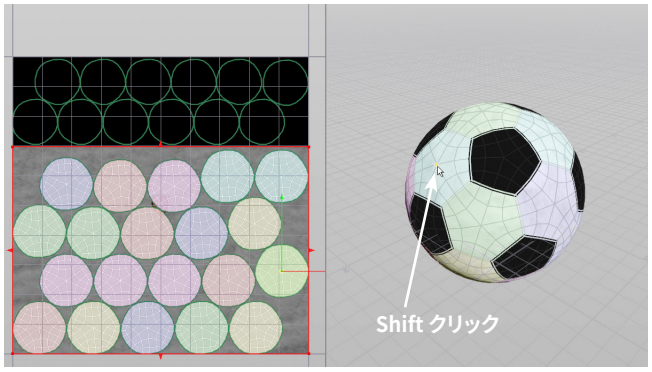
04

uvflatten ノードの後、Fuse ノードの前に、もう 1 つ **UV Flatten** ノードを追加します。Group フィールドで、`@patches<20` というエクスプレッションを入力します。これにより、明るいパッチが平坦化されます。

Handle ツール を **RMB クリック**し、***Min:*** ハンドルをオンにします。上部にある矢印ハンドルを、すべてのパッチがテクスチャマップ下部の明るい領域内に入るまで引き下げます。

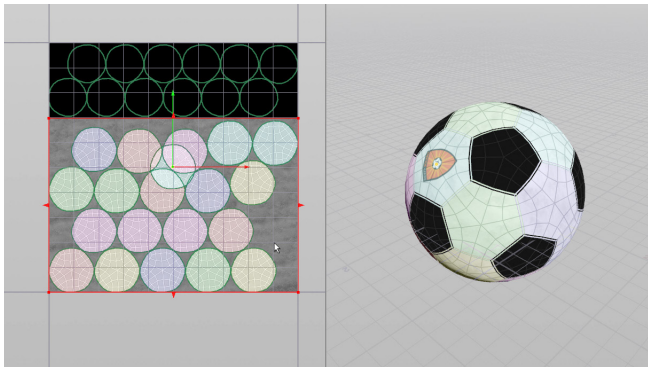
完了したら、Handle ツール を **RMB クリック**し、***Min:*** ハンドルをオフにします。

Scene View で RMB クリックし、メニューで **Texture Visualization > Off** を選択します。これでグリッドが非表示になります。



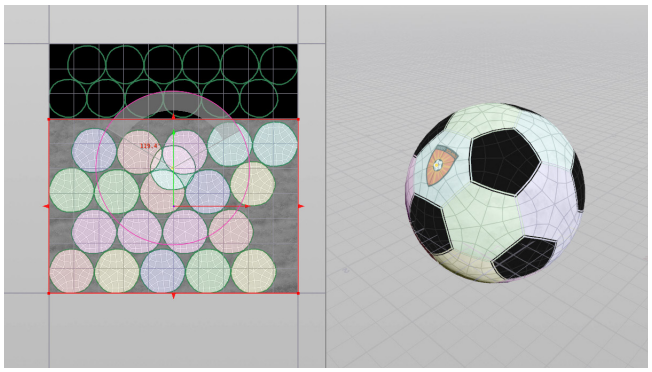
05 ***uvflatten*** ノードを選択し、**Handle** ツールをクリックします。ジオメトリ上にカーソルを置くと、パッチがハイライトされるのが分かります。ビューポート上部にある**オペレーションコントロール**ツールバーの **Pin Vertices** ボタンをクリックします。

この画像に表示されたパッチにカーソルを置き、3D ビューで、目的のパッチの中央の頂点を **Shift クリック**して選択します。UV ビューに、ピンと操作用のハンドルが追加されます。

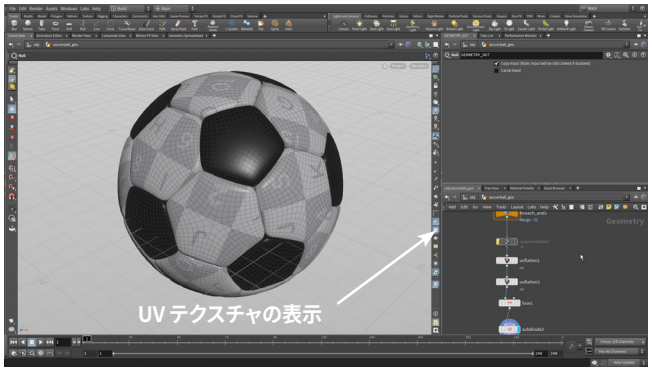


06 UV ビューに**移動**し、ピンとパッチをロゴの中央になるように移動します。**Y を押して**回転ハンドルに切り替え、パッチを回転してロゴの位置を調整します。

UV ビューでパッチの頂点をピン留めすると、位置を固定できます。最初は隣接するパッチとのオーバーラップがいくらかありますが、再パックによって簡単に修正できます。



07 **オペレーションコントロール**ツールバーの **Repack** ボタンをクリックして、新しいパッチを中心に残りのパッチを再編成します。引き続きピンを使用してパッチを動かせますが、UV のオーバーラップを回避するには、もう一度**再パック**が必要です。



08 ***quickshade*** ノードを**バイパス**に設定し、テクスチャマップの割り当てを非表示にします。このノードが必要なのは、UV をセットアップするときのみです。サッカーボールに UV グリッドが表示されます。

パースビューで UV を非表示にするため、**Display Options** バーで **Show UV Texture when UV's Present** ボタンを**オフ**にします。

Null ノードをチェーンの終端に追加して、名前を ***GEOMETRY_OUT*** に変更します。ネットワークチェーンの終端の定義には、この種のノードを使うことをお勧めします。作業内容を**保存**します。

UV FLATTEN

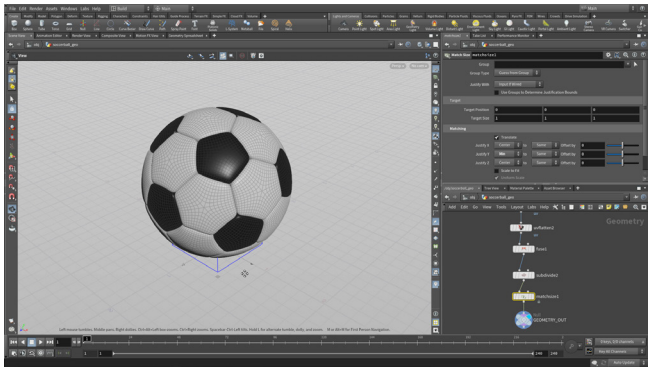
UV Flatten ノードの動作は、2 つの工程で構成されています。シーム（継ぎ目）によって定義された、個々のテクスチャのピースを受け取り、それらを 2D テクスチャ空間へと平坦化して、ポリゴンサイズの均等化を試みます。

このノードで、平坦化アルゴリズム用の拘束を追加することができます。この拘束により、レイアウトアルゴリズムが強制的に特定の条件を満たすようにすることで、最終的な UV レイアウトをさらに制御できるようになります。このノードの状態で、ツールを使って拘束を指定すれば、ノードをインタラクティブに使用できます。または、Enable Manual Layout をオフにすると、このノードをプロシージャルに使用できるようになります。

パート 5

レイアウト：カメラとライト

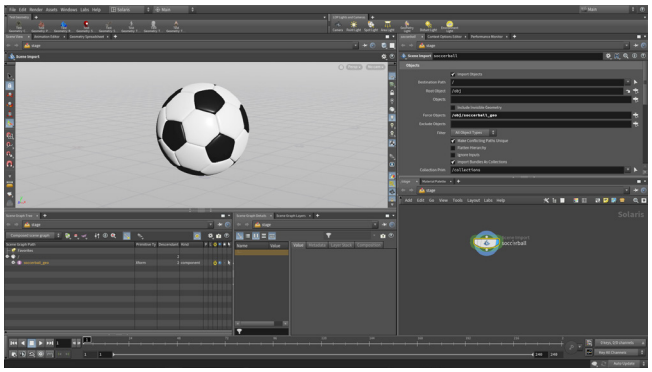
レンダリング用のシーンを作成するには、ジオメトリを Houdini の Solaris または LOP コンテキストに取り込むことになります。これは、ルックデブ、レイアウト、ライティングに特化した環境で、USD (Universal Scene Description) を基盤に構築されています。これを使用すると、Solaris ワークフローの一部として、Scene View で適切に動作する Karma レンダラでレンダリングすることができます。



01

ネットワークビューで、**Tab > Match Size** を押し、***subdivide*** と ***GEOMETRY_OUT null*** の間にこのノードを追加します。***GEOMETRY_OUT*** ノードに **Display フラグ**を設定します。

matchsize ノードを選択し、**Justify Y** を **Min** に設定して、地上に乗るようにボールを引き上げます。これで、正しい位置でレンダリングできます。



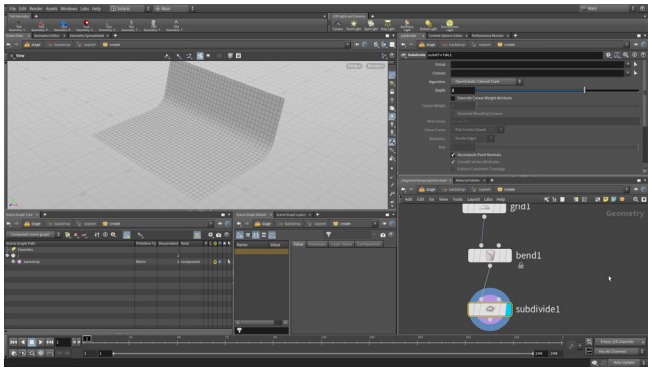
02

デスクトップを **Solaris** に変更します。パスバーに **stage** と表示されていることを確認してください。

ネットワークビューで **Tab > Scene Import** を選択し、クリックしてノードを配置します。**Force Objects** フィールドの横にある***ノードセレクタ***ボタンをクリックし、ポップアップウィンドウで ***soccerball_geo*** オブジェクトを選択してから **Accept Pattern** をクリックします。

Scene View で、**スペースバー + H** など、ビューのリセット用ツールを使用して、サッカーボールがよく見えるようにします。

注：Objects フィールドではなく Force Objects を使用します。Force Objects は、オブジェクトの Display フラグがオフになっていても、LOP にオブジェクトを取り込むからです。

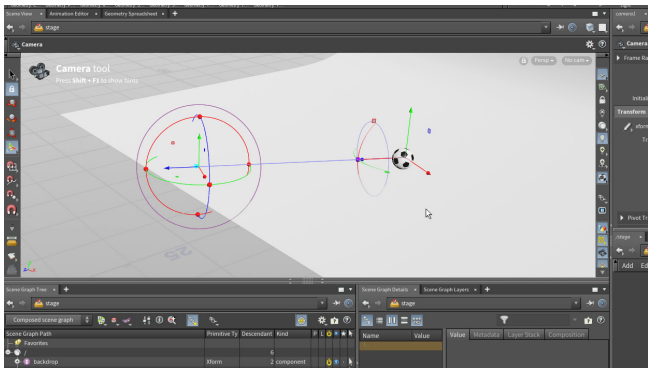


03

ネットワークビューで、**Tab** を押して **Grid** と入力します。クリックしてノードを配置し、名前を ***backdrop*** に変更します。***backdrop*** ノードをダブルクリックして、ジオメトリレベルに入ります。

grid ノードを選択し、Size を **80, 80**、**Center** を **0, 0, -20** に設定します。***grid*** ノードの出力を **RMB クリック**して、**Bend** と入力します。クリックしてノードを配置したら、**Display フラグ**を設定します。**Bend** を **75**、**Capture** セクションで **Capture Origin** を **0, 0, -30**、**Capture Direction** を **0, 0, -1**、**Capture Length** を **5** に設定します。

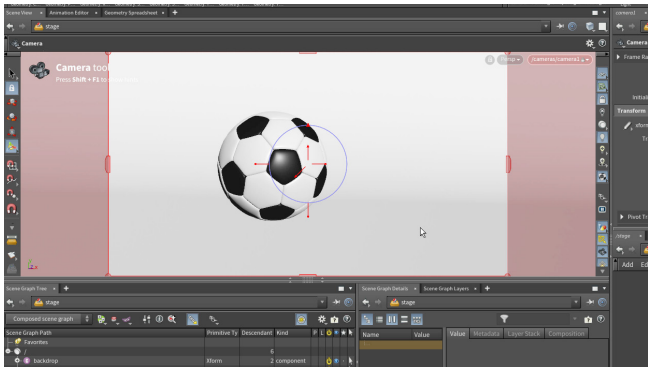
grid ノードの出力を **RMB クリック**して、**Subdivide** と入力します。**Display フラグ**を設定し、**Depth** を **2** に設定します。



04

ステージレベルに戻ります。***backdrop*** ノードを ***sceneimport*** に接続します。sceneimport の出力を **RMB クリック**し、**Camera** と入力します。**Enter** を押してノードを配置したら、その **Display フラグ**を設定します。

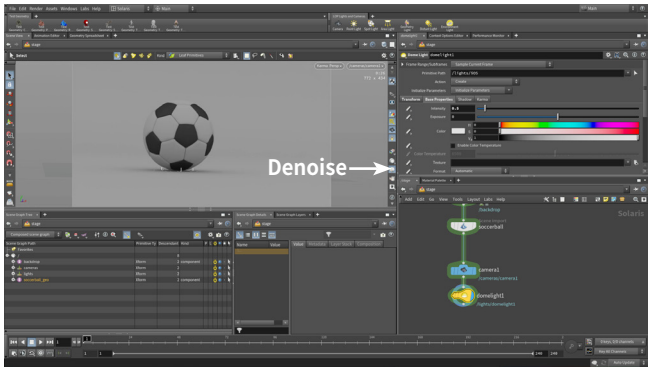
Scene View で、原点にカメラハンドルが表示されます。ズームアウトしてシーンを見下ろすようにしたら、カメラが左側からサッカーボールを見るようにハンドルを調整します。地面に沿ってハンドルを動かしたい場合には、**コンストラクション平面**をアクティブにすると操作がしやすくなります。また、軸ハンドルで方向を制御したり、カメラを地面から持ち上げることも可能です。



05 **Scene View** の右上の ***No cam*** メニューをクリックし、***camera1*** を選択します。これでカメラ越しに見えるようになり、見え方を調整できます。

求めているのはこのビューではないでしょう。いくらか変更します。Scene View の右側にある **Lock camera to view** ボタンをクリックします。**表示ツール（スペースバー + LMB/MMB/RMB）**を使用してカメラの位置を調整します。

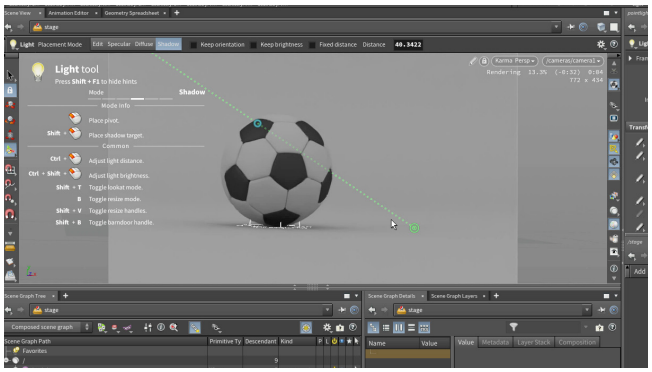
重要：完了したら、**Lock Camera** ボタンをオフにします。



06 **LOP Lights and Camera** シェルフで **Environment Light** ツールをクリックし、**Enter** を押してそれを原点に配置します。**Intensity** を **0.5** に設定し、少し和らげます。

カメラメニューの左にあるメニューをクリックし、**Karma** に設定します。これで、ビューポートで Karma レンダラが使用されるようになります。

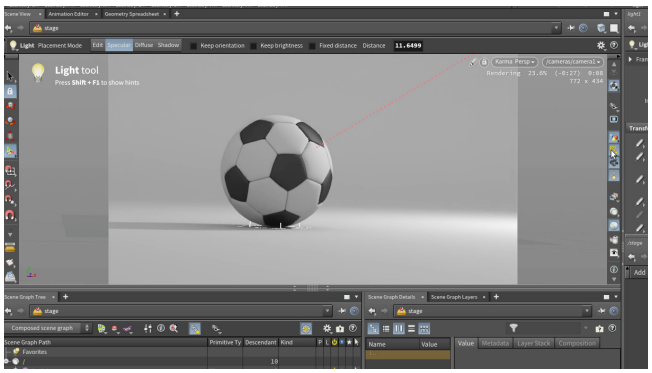
Nvidia グラフィックカードがあり、最新のドライバをインストールしている場合は、Optix Denoiser をオンにすると、画像をより素早く処理できます。**Display Options** バーでそれをオンにするか、**D を押して Render Display Options** の **Enable Denoising** を設定します。



07 **LOP Lights and Camera** シェルフで **Point Light** ツールを**クリック**し、**Enter** を押してそれを原点に配置します。ライト越しに見えるようになります。カメラを再度 ***camera1*** に設定します。

ノードをアクティブにした状態で、**Shift + F** を押して **Shadow** モードをオンにします。**オペレーションコントロール**ツールバーでクリックしても同じです。サッカーボールの上部を**クリック**してピボットポイントを設定したら、**Shift クリック**して地面にターゲットを配置します。**Ctrl ドラッグ** でライトの距離を設定します。

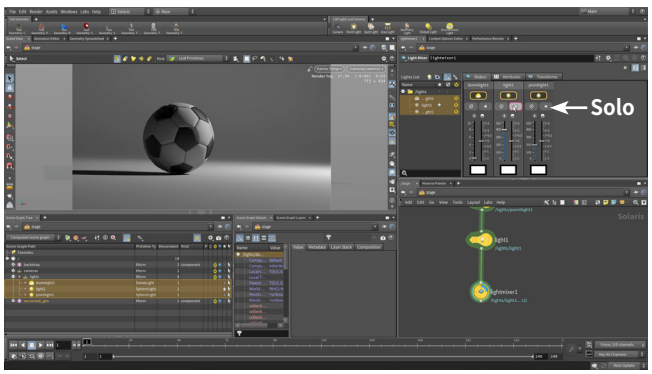
Ctrl + Shift ドラッグで、ライトの強度を変更できます。サッカーボールですぐに見て取れるほどの影響を与えるには、かなり高く設定する必要があります。



08 ネットワークビューで、***pointlights*** ノードの出力を **RMB クリック**し、**Light** と入力してから **Enter** を押してノードを配置し、その **Display フラグ**を設定します。

ノードをアクティブにした状態で、**Shift + S** を押して **Specular** モードをオンにします。サッカーボールの右側をクリックして、鏡面反射が集中する領域を定義します。

Ctrl ドラッグでライトをサッカーボールから離し、**Ctrl + Shift ドラッグ**でライトの強度を変更します。



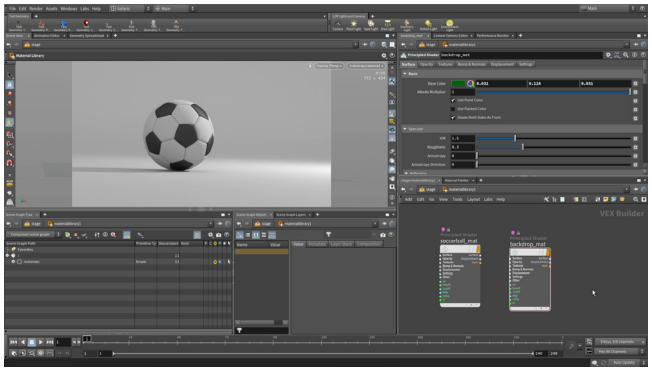
09 ネットワークビューで、***lights*** ノードの出力を **RMB クリック**し、**Light Mixer** と入力してから **Enter** を押してノードを配置し、その **Display フラグ**を設定します。パラメータエディタに特殊なパネルが作成され、左側にライトがリストされます。

3 つのライトをリストから右側の領域に**ドラッグ**します。**星**アイコンをクリックして各ライトを**ソロ**にして寄与度を定めたら、**Exposure** でライティングを微調整します。**intensities** がとても高いので、強度バーの上のアイコンをクリックし、ポップアップメニューでショットに適した **Max Value** を設定します。完了したら、**Solo** ボタンを**オフ**にしてすべてのライトを表示します。

パート 6

ルックデブ：マテリアル

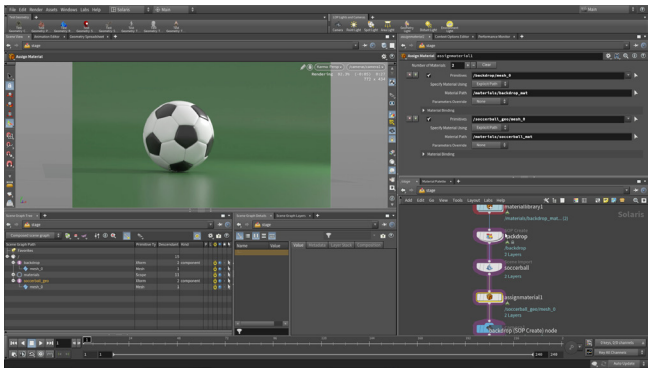
マテリアルとシェーダは、LOP/Solaris のコンテキスト内で作成することもできます。この場合、シーングラフにマテリアルを追加してから、ジオメトリに割り当てます。マテリアルは、Material Library ノード内で作成され、LOP/Solaris レベルで割り当てられます。背景にテクスチャを追加するには、UV を作成して、マップを適切に配置できるようにする必要があります。



01 ネットワークビューで **Tab > Material Library** を押し、既存のノードのネットワークのすぐ上にノードを配置したら、それを ***backdrop*** ノードに接続します。ノードを**ダブルクリック**して、**VEX Builder** レベルに下がります。

Tab > Principled Shader を押してノードを配置します。その名前を ***soccerball_mat*** に変更します。**Base Color** を **白 (1, 1, 1)** に設定します。

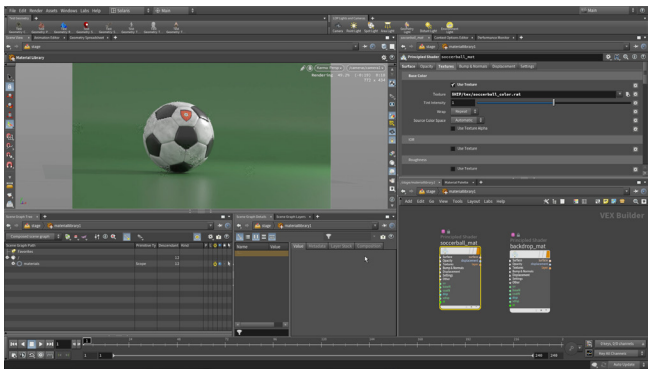
このノードを **Alt ドラッグ**して、2 つ目の Principled Shader を作成し、名前を ***backdrop_mat*** に変更します。**Base Color** を暗い緑色に変更します。



02 **ステージ**レベルに戻り、**Tab > Assign Material** を押して、***sceneimport*** と ***camera*** ノードの間にノードを配置します。

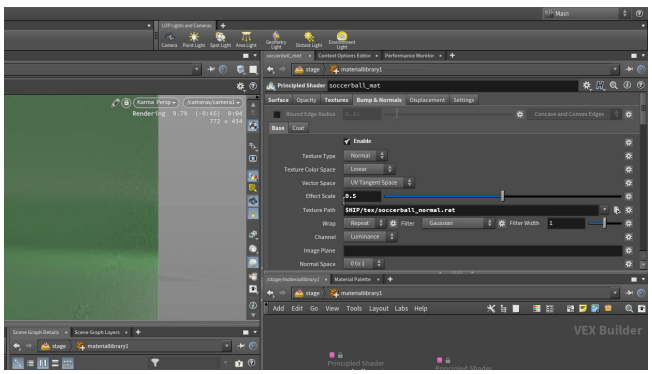
パラメータエディタで、**Primitives** の横の**矢印**ボタンをクリックし、ビューポートで ***soccerball*** ジオメトリを選択します。**Enter を押し**、**Primitives** フィールドにそのパスを追加します。**Material Path** の横にある**矢印**をクリックし、ポップアップウィンドウで ***materials > soccerball_mat*** を選択してから **OK** をクリックします。

+ (プラス) ボタンをクリックし、***backdrop*** と ***backdrop_mat*** についてもこの手順を繰り返します。



03 ***materiallibrary*** ノードをダブルクリックして中に入り、***soccerball_mat*** ノードを選択します。**Textures** タブをクリックし、**Base Color** の **Use Texture** をクリックしたら、**Texture** の横のボタンを使用してファイルウィンドウを開きます。横のリストで **$HIP** をクリックしたら、***tex*** フォルダをクリックして開き、***soccerball_color.rat*** をワンクリックして選択します。**Accept** をクリックし、テクスチャをマテリアルに割り当てます。

$HIP 参照により、シーンファイルの位置に対して相対的に参照するようになります。このようにすると、プロジェクトディレクトリを別のコンピュータに移動しても、引き続き参照が機能します。



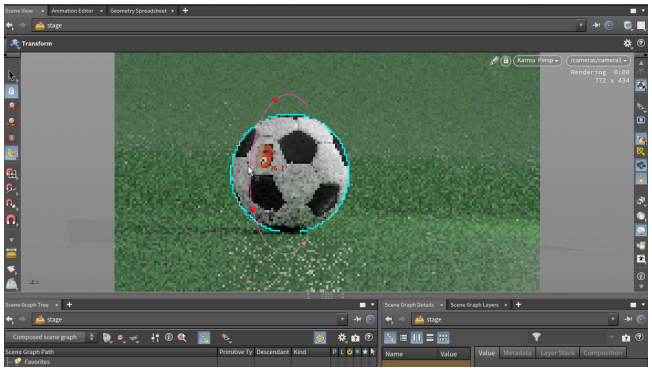
04 **Textures** タブで、前の手順で学んだテクニックを使用して、**Roughness** と **Reflectivity** にテクスチャを割り当てます。***tex*** フォルダで適切なテクスチャを見つけます。

マテリアルの **Bumps & Normals** タブに移動して、**Enable** ボタンをクリックします。**Texture Path** の横にある矢印をクリックして、tex ディレクトリから ***soccerball_normal.rat*** ファイルを選択します。**Effect Scale** を約 **0.5** に設定し、どうなるかを確認します。

Houdini のマテリアル

Houdini のマテリアルは **VEX Builder** コンテキストにあり、この場合は **Material Library** ノード内にネストされます。マテリアルは、マテリアルの品質を定義する **VOP** ノードまたは **Material X** ノードで構成されています。

Principled Shader は、単体でテクスチャマップを割り当て、多様なルックを生成できる優れたマテリアルです。独自のシェーダやマテリアルを構築して、より高度なルックを実現することも可能です。

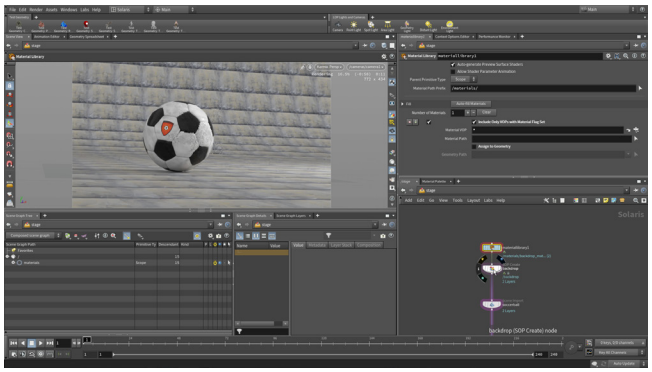


05

Scene View で、***camera1*** 越しにショットを確認します。サッカーボールのロゴがカメラに対してよい位置にないので、ボールを回転する必要があります。

Scene View で、サッカーボールを選択します。**Tab > transform を押します**。チェーンの終端に transform ノードが追加されます。**Handle** ツールで、**R を押して Rotate** ハンドルにしたら、ロゴがきれいに見えるようにボールを回転します。タンブルする必要がある場合もあります。

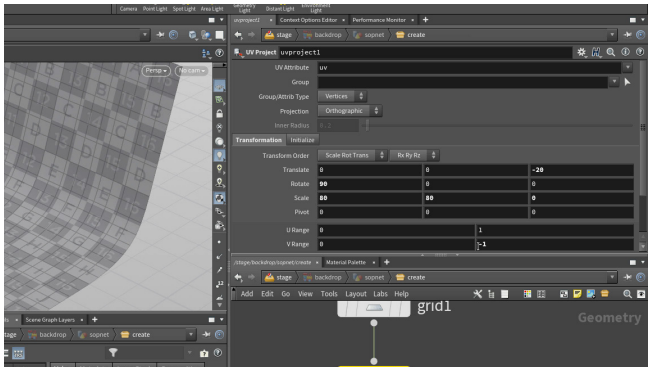
これは、**Karma** または **Houdini GL** ビューで行うことができます。変更が気に入らない場合は、ノードを削除してもう一度試します。***soccerball_geo*** を選択解除するには、**シーングラフ** でそれを **Ctrl** クリックします。



06

backdrop マテリアルにいくつかテクスチャマップを追加しましょう。***materiallibrary*** ノードを**ダブルクリック**して、***backdrop_mat*** ノードを選択します。

Base Color を **1, 1, 1** に設定し、この色がテクスチャマップに乗算されるようにします。**Textures** タブをクリックして、**Base Color** の **Use Texture** をクリックします。**ファイル選択**ボタンをクリックして、`$HIP` から `/tex` ディレクトリに移動し、***backdrop_color.rat*** を選択します。***Reflectivity*** に **backdrop_reflect.rat** テクスチャを追加してもよいでしょう。**Scene View** を見ると、UV が適切にセットアップされておらず、テクスチャマップも機能していないことが分かります。

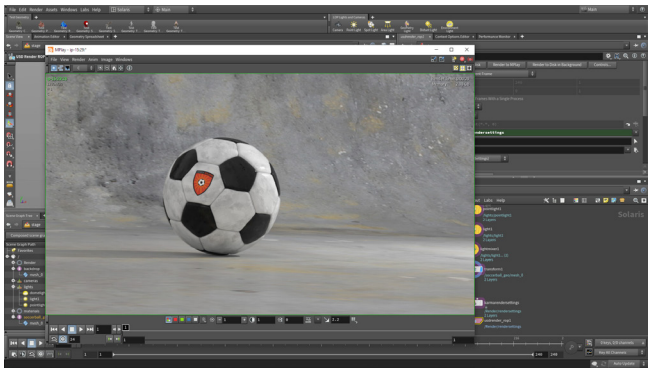


07

ステージレベルに戻り、***backdrop*** ノードを**ダブルクリック**してジオメトリレベルに移動します。***grid*** と ***bend*** の間に **UV Project** ノードを追加します。サーフェスを曲げる前に UV が作成されるよう、ここにノードを追加していきます。

uvproject ノードで、**Initialize** タブをクリックして **Initialize** ボタンを押します。**Transformation** タブに戻り、**V Range** を **0, -1** に設定します。これにより、UV の方向が適切になります。

重要 : subdivide ノードに **Display フラグ**を再度設定します。



08

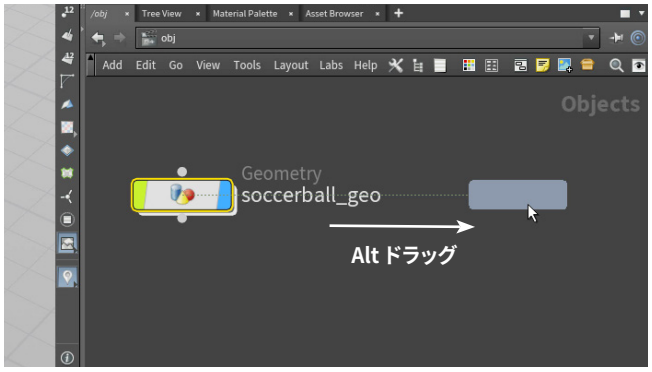
ステージレベルに戻ります。**ネットワークビュー**で、**Tab > Karma** を押し、**Karma Render Settings** と **USD Render ROP** ノードを追加します。それらをチェーンの終端に接続します。***karmarendersettings*** ノードを選択して、**Image Output > Filters** タブで **Denoiser** を **nvidia Optix Denoiser** に設定してデノイザをオンに戻します。

usdrender_rop ノードを選択します。**Render to Mplay** ボタンをクリックします。**Mplay** が開き、進行中のレンダリングが表示されます。**File > Save Frame As** を選択し、画像をディスクに保存します。

パート 7

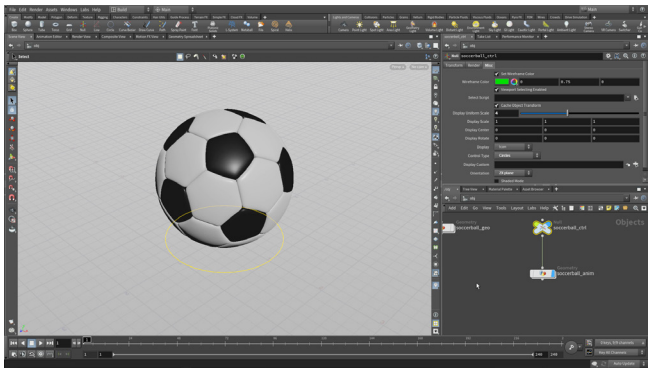
サッカーボールのリギング

バウンシングボールのアニメーションを作成するには、シンプルなリグの作成からはじめます。これで、キーフレームが設定しやすくなります。そのためには、ビューポートでインタラクティブに作業できるように、Null オブジェクトをセットアップします。また、サッカーボールのジオメトリネットワークにノードを追加して、ボールの回転、スクワッシュとストレッチを適用します。



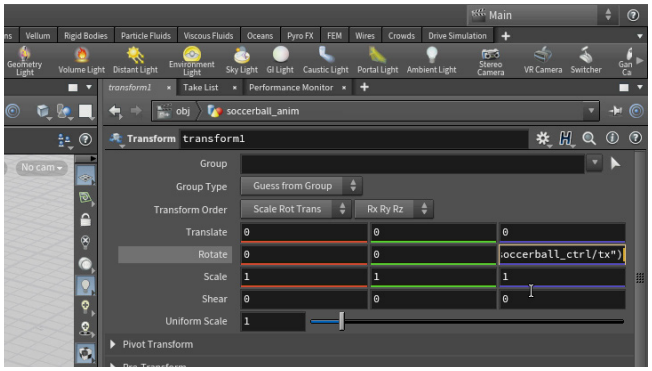
01 **Build** デスクトップに戻り、パスバーの 1 つをクリックしてオブジェクトレベルに移動し、***obj*** を選択します。ネットワークエディタで、***soccerball_geo*** を **Alt ドラッグ**してコピーを作成します。このノードの名前を socccerball_anim に変更します。

soccer_anim を使用して、リグを構築します。***soccer_geo*** ノードの Display フラグをオフにして、非表示にします。元のセットアップは変更したくありません。そのオブジェクトは、Solaris コンテキストのショット 1 で使用されているからです。この新しいサッカーボールは、アニメートされるショット 2 で使用します。



02 **Create** シェルフで、**Null** ツールをクリックしてから **Enter** を押し、それを原点に配置します。名前を ***soccerball_ctrl*** に変更します。**Misc** タブに移動し、**Control Type** を **Circles** に、**Orientation** を **ZX Plane** に設定します。**Display Uniform Scale** を **4** に設定します。これでリグのハンドルが作成されます。簡単に選択でき、後のレンダリング工程ではレンダリングされません。

ネットワークエディタで、***soccerball_anim*** オブジェクトの入力を ***soccerball_ctrl*** Null の出力に接続し、親子関係を作成します。Null を動かすと、ボールも動くようになります。***soccerball_anim*** の**選択フラグ**をオフにして、アニメーションの間ビューポートで誤って選択しないようにします。代わりに ***soccerball_ctrl*** を使用します。

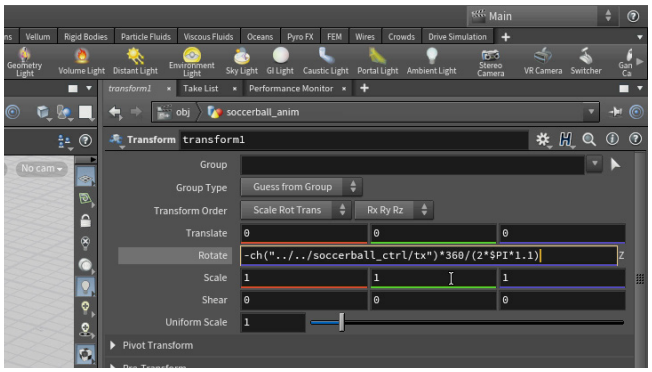


03 ***soccerball_ctrl*** ノードを選択します。パラメータエディタで、**Transform** タブをクリックしてから **Translate X** パラメータを **RMB クリック**します。**Copy Parameter** を選択します。

soccerball_anim オブジェクトの中に入ります。**subdivide** と ***matchsize*** ノードの間に ***Transform*** ノードを追加します。**Rotate Z** を **RMB クリック**し、**Paste Relative References** を選択します。このパラメータに、チャンネル参照のエクスプレッションが配置されます。

```
ch("../../soccerball_ctrl/tx")
```

これで、コントロールオブジェクトの動きがこのノードの回転とつながるようになります。

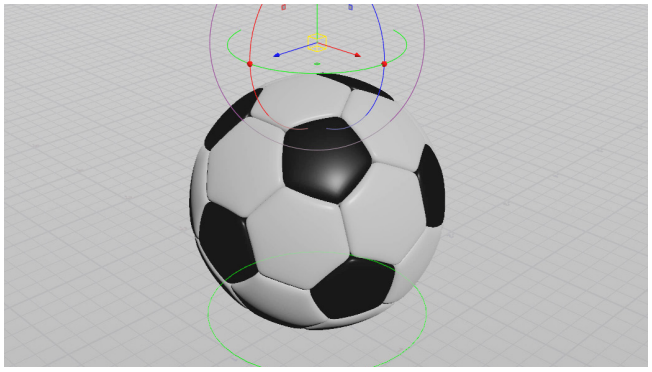


04 パラメータをクリックしてチャンネルを展開します。**ボールの外周 (2πr)** を使用して、ボールが前進するときの回転を定めます。

エクスプレッションを次のように変更します。

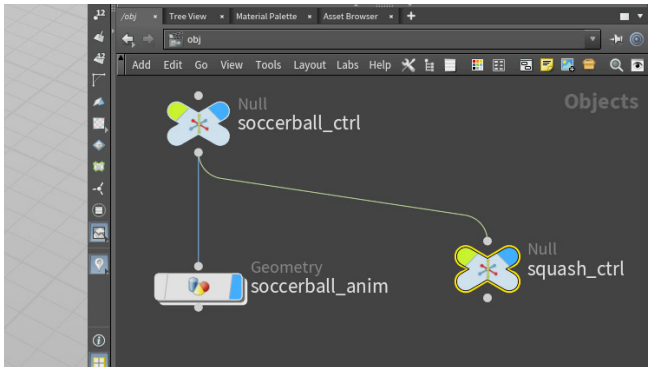
```
-ch("../../soccerball_ctrl/tx")*360/(2*$PI*1.1)
```

まず、先頭にマイナス **(-)** 記号を追加します。その後、エクスプレッションでボールの位置に **360** 度を乗算し、**2πr** (π は **$PI**) で除算します。オブジェクトレベルで、***soccerball_ctrl*** を **X 軸**に沿って動かします。エクスプレッションにより、ボールが動きに合わせて回転します。完了したら、ボールを原点に戻します。



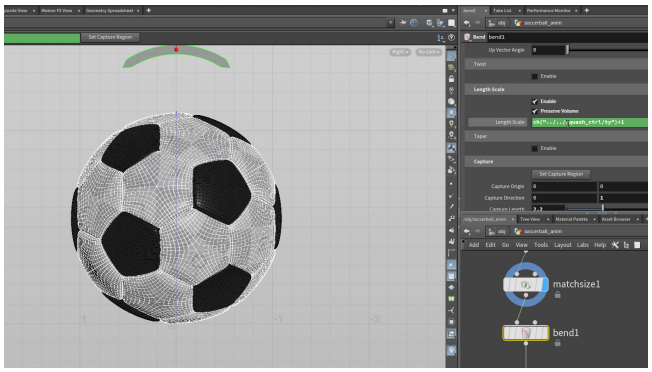
05 ビューポートで、もう 1 つ **Null** オブジェクトを原点に作成します。名前を ***squash_ctrl*** に変更します。**Misc** タブに移動して、**Control Type** を **Box** に、**Display Uniform Scale** を **0.2** に設定します。

Null をボールの真上に移動します。**Translate Y** は約 **2.5** になるはずです。パラメータエディタで、**Modify Pre-Transforms** メニューを選び、**Clean Translates** を選択します。これにより、Null が地面より上にあっても、その **Translate Y** 値が **0** に設定されます。Null でスクワッシュとストレッチを駆動するためには、0 のデフォルト値が必要となります。



06 ***soccerball_ctrl*** Null を ***squash_ctrl*** Null の親にします。これにより、コントロール Null をアニメートすると、このセカンダリの Null が動くようになります。

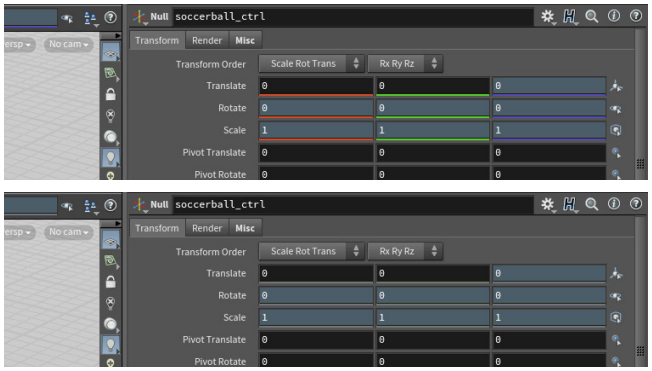
squash_ctrl ノードの **Translate Y** パラメータを **RMB クリック**します。**Copy Parameter** を選択します。このパラメータを使用して、ボールのスクワッシュとストレッチを駆動します。こうすることで、ビューポートからインタラクティブにスクワッシュとストレッチを制御できるようになります。



07 ***soccerball_anim*** オブジェクトの中に入ります。***matchsize*** の後に **Bend** ノードを追加します。**Limit Deformation to Capture Region** チェックボックスを**オフ**にします。

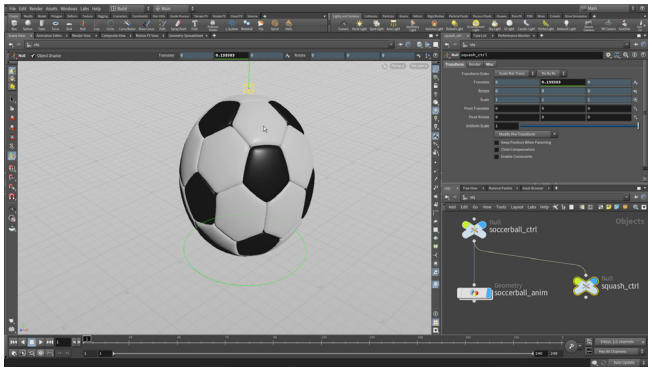
Right ビューに移動して **Set Capture Region** ボタンをクリックします。**グリッドスナッピング**をオンにして、ボールの底部と上部に 1 つずつポイントを配置します。これにより、**Up Vector** は **0, 0, 1**、**Capture Direction** は **0, 1, 0**、**Capture Length** は **2.2** に設定されるはずです。

Length Scale と **Preserve Volume** をオンにしたら、**Length Scale** を **RMB クリック**して、**Paste Relative References** を選択します。エクスプレッションの最後に **+1** を追加します。



08 オブジェクトレベルに移動し、パースビューを表示します。**Transform X** および **Transform Z** パラメータを **RMB クリック**し、**Lock Parameter** を選択して、***squash_ctrl*** でこれらのパラメータをロックします ***Scale*** および ***Rotate*** パラメータを **RMB クリック**し、**Lock Parameter** を選択して 3 つすべてのチャンネルをロックします。

soccerball_ctrl オブジェクトを選択します。**Translate X** と **Translate Y** を除くすべてのチャンネルを**ロック**します。これでコントロールを選択すると、ロックされていないチャンネルのハンドルのみが表示されるようになります。アニメータは選択したパラメータのみを操作でき、リグが扱いやすくなります。



09 リグを X と Y で動かして、テストしてみましょう。2 つ目のハンドルを使用して、スクワッシュとストレッチを適用します。すべてのパーツがうまく機能していることを確認したら、すべての値を 0 に戻し、アニメーションの準備をします。

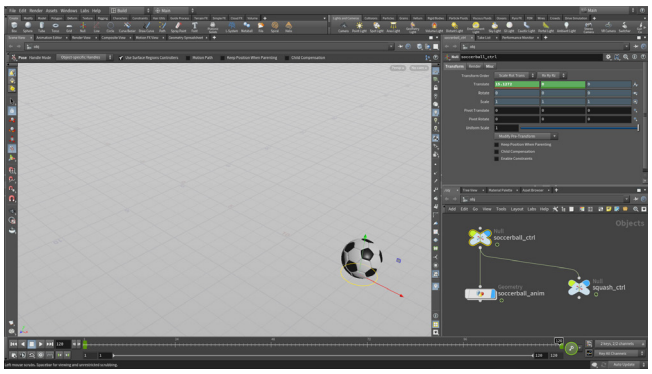
Scene View の左側のツールバーで **Secure Selection** をオフにすることをお勧めします。**Move** ツールを使用している間、2 つのコントロール Null を選択しやすくなります。オフにしない場合は、選択を切り替えるたびに **S キー**を押す必要があります。

シーンファイルを**保存**してから、次に進みます。

パート 8

バウンシングボールのアニメーション

サッカーボールのリグを使用して、バウンドする（弾む）ボールをアニメートします。キーフレームを設定したり、アニメーションカーブを調整したり、ビューポートでハンドルを使って速度を調整する方法を学びます。バウンシングボールは、アニメーションの練習によく使われる課題で、Houdini でのアニメーションの基礎を学ぶのにも最適です。

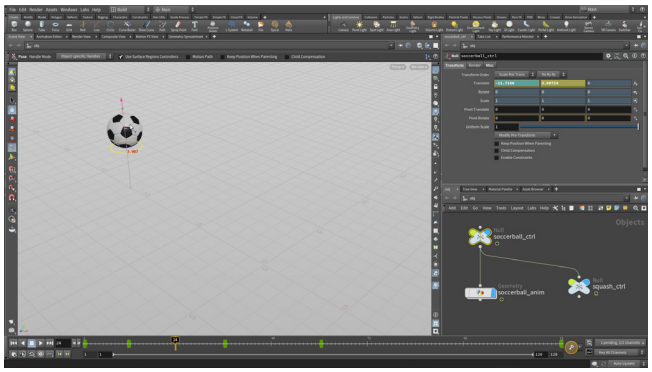


01

タイムラインの左端の下段で、**Global Animation Options** ボタンをクリックします。***End*** を **120** に設定し、**Close** をクリックします。これにより、タイムラインの範囲が 120 フレームに設定されます。

フレーム 1 にいることを確認します。左側のツールバーで **Pose** ツールをクリックし、***soccerball_ctrl*** を選択します。ボールを X 方向の約 **-15** に移動したら、**K** を**押して**キーフレームを設定します。タイムラインを**フレーム 120** に移動します。ボールを X 方向の約 **15** に移動したら、**K** を**押して** 2 つ目のキーフレームを設定します。

タイムラインを**スクラブ**し、ボールがアニメートされていることを確認します。リグの設計に基づいて、移動および回転するはずです。

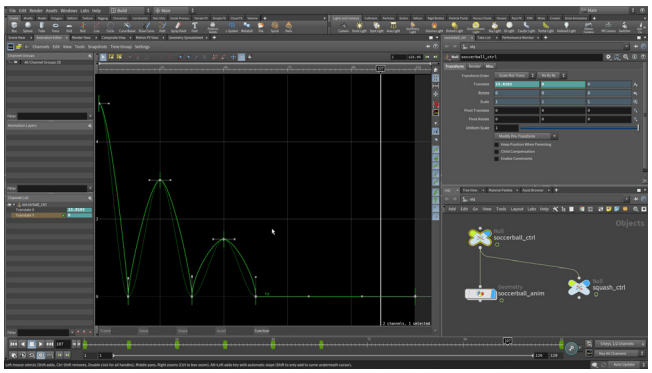


02

タイムラインをフレーム **12** に移動し、**K を押して**中間のキーを設定します。フレーム **36** と **60** で繰り返します。これらのキーフレームは、すべて地面上にあります。

次に**フレーム 1** に移動し、ボールを Y 方向に上げます。もう 1 つキーを設定する必要はありません。この移動によって、フレーム 1 に既に設定してあるキーフレームが更新されるからです。

フレーム 24 に移動し、ボールをフレーム 1 よりも少し低い位置まで Y 方向に上げます。**K を押して**キーフレームを設定します。**フレーム 48** に移動し、ボールをさらに少し低い位置まで上げます。**K を押して**もう 1 つキーフレームを設定します。



03

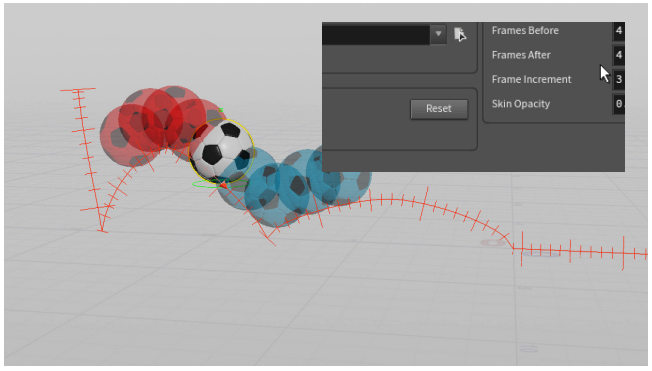
タイムラインを**スクラブ**すると、ボールが浮いたように見えますが、ボールが着地するときは、強く当たるようにしたいわけです。**Animation Editor** ペインタブをクリックします。

Scoped Parameters リストで ***Translate Y*** チャンネルをクリックします。**H を押して**、カーブのビューをホーム（リセット）します。ボールが着地する 3 つのキーフレームを選択し、グラフのすぐ上にある **Function** ツールバーで **Untie Handles** ボタンを押します。何もない空間をクリックして選択を解除したら、接線ハンドルを微調整して、着地の各ポイントのカーブを鋭角にします。また、上部のハンドルは平坦にして、バウンドのピークでボールの速度を遅くします。

Pose ツール

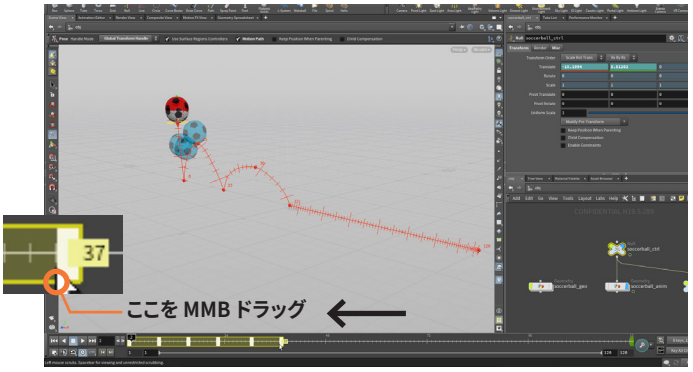
Move または **Handle** ツールを使用すると、サッカーボールのリグの 2 つのコントロールオブジェクトを簡単に操作できます。**Pose** ツールを使用する利点は、**Motion Path** ハンドルにアクセスでき、インバースキネマティクスを使用している場合は、専用のハンドルを使用できることです。そのため、リグにキーフレームを設定する場合は、このツールを必ず思い出してください。**Secure Selection** がオンになっていても、**Pose** ツールで別のオブジェクトを選択可能です。

- **Select**
- **Secure Selection [~]**
- **Move [T]**
- **Rotate [R]**
- **Scale [E]**
- **POSE [Y または Ctrl + R]**
- **Handle [Enter]**



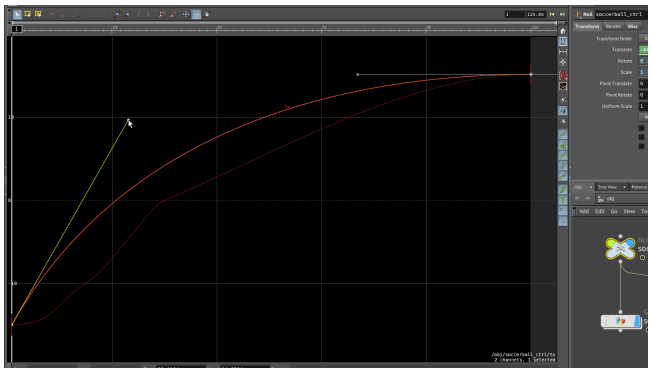
04 **Scene View** に戻り、結果をプレビューします。ボールが着地するたびに、弾むようになりました。**Pose** ツールがアクティブで、***soccerball_ctrl*** ノードが選択されていることを確認します。上部のバーで、**Motion Path** ハンドルを**オン**にします。これにより、ボールのバウンドのパスが表示されます。それぞれのキーフレームマーカーをクリックして、弾み方を微調整します。カーブをより詳細に制御するには、ハンドルを **RMB クリック**して **Show Tangents** をオンにします。

soccerball_anim オブジェクトの **Misc** タブに移動して、**Onion Skinning** を **Full Deformation** に設定します。**スペースバー + D** を押し、**Scene** タブで **Frame Increment**、**Frames Before**、**Frames After** のカラーを調整します。



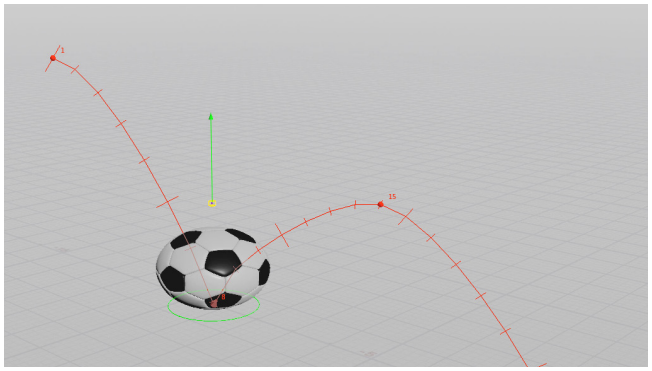
05 タイミングの調整には、タイムラインも使用できます。**Shift** を押し、フレーム 1 からタイムラインの最後のキーまで境界ボックスをドラッグして、すべてのキーを**選択**します。次に、ボックスの端のハンドルの下を **MMB ドラッグ**して、バウンドのタイミングをスケールし、スピードを上げます。**MMB** を使用して各キーを選択してから、**MMB** でドラッグしても、それぞれのキーフレームのタイミングを好きなように調整できます。

ここでは、バウンドのタイミングを決定します。希望する動き方になるまで試してください。Translate X の値が原因で、弾み方が自然ではないかもしれません。それは次の手順で修正します。



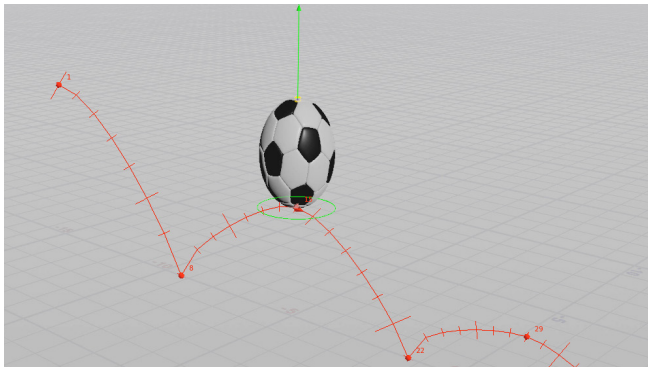
06 **Animation Editor** ペインタブをクリックして、2 つのカーブを表示します。Scoped Parameters リストで、***soccerball_ctrl*** の **Translate X** をクリックします。最初と最後のキーを除く、すべてのキーを選択します。**Delete** を押します。カーブハンドルを使用して、急な傾斜が緩い傾斜に移行するようにします。こうすると、ボールの動きは始めが速く、終わりでゆっくりになります。

Motion Path ハンドルで X 方向のポイントを再び調整すると、その方向には中間キーがないため、おかしな結果になることに注意してください。これ以降は、Y 方向の微調整にのみこのハンドルを使用します。

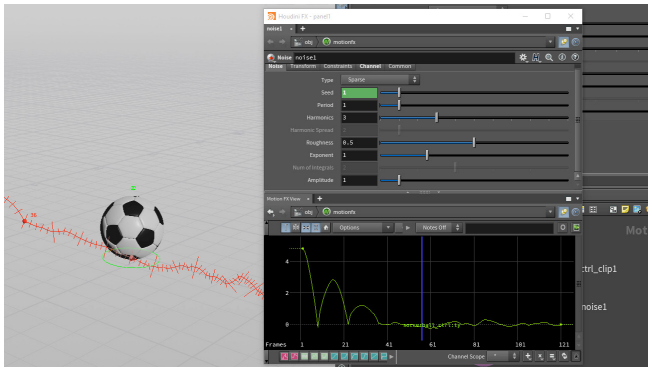


07 Scene View に戻ります。**Motion Path** ハンドルを **RMB クリック**し、**Persistent** を選択します。これにより、スクワッシュとストレッチにキーフレームを設定する際のガイドとして、ハンドルを保持しておけます。

squash_ctrl を選択し、**Motion Path** をオフにします。最初のバウンドに移動して、1 フレーム戻ります。***squash_ctrl*** のハンドルを選択し、ボールを少し引き伸ばし（ストレッチし）ます。**K** キーをで**キーフレームを設定**します。バウンドのフレームに移動し、ハンドルを下げてスクワッシュ（潰し）を作成します。**もう 1 つキーを設定します。**1 フレーム進め、ボールが丸くなるまでストレッチし（引き伸ばし）ます。**もう 1 つキーを設定します。**この手順をすべてのバウンドで繰り返します。

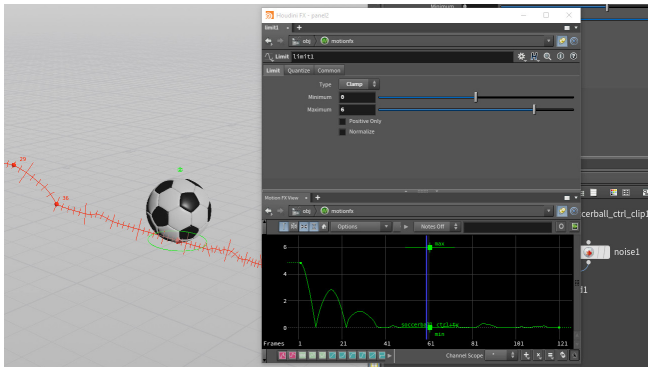


08 完了したら、モーションをスクラブして再生し、結果をプレビューします。バウンドのピークで、ボールが引き伸ばされていることを確認してください（ストレッチ）。モーションを正しく評価するには、タイムラインで **Real Time Toggle** が**オン**になっていることを確認します。次に **Animation Editor** を使用して、スクワッシュとストレッチを微調整します。キーフレームとサッカーボールのバウンドが揃っていることを確認しましょう。



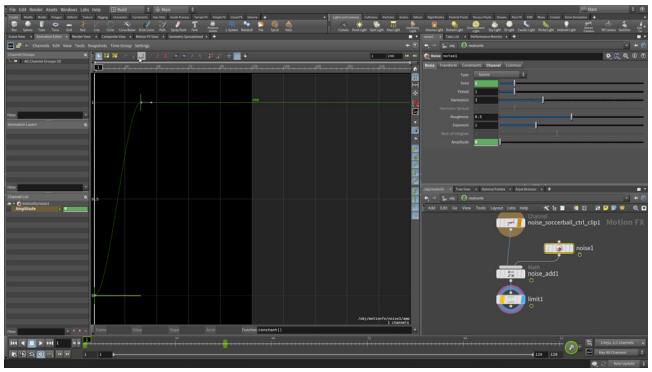
09 ***soccerball_ctrl*** Null オブジェクトを選択します。**Translate Y** を **RMB クリック**し、**Motion FX > Noise** を選択します。パネルが開き、そこに表示されたパラメータを使用してノイズを制御できます。***soccerball_ctrl*** の **Translate Y** チャンネルに情報を送り返す CHOP ノードを含む、新しいサブネットワークが作成されます。

Amplitude を **5** に設定し、**Play** を押してどのように見えるかを確認します。上下に大きく動き、まるで激しい乱流のようです。**Amplitude** を **1** に設定して、ボールの動きに加わる揺れを少なくします。



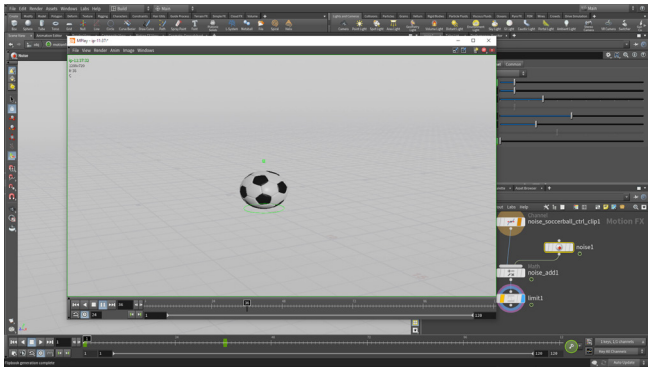
10 動きの一部が地面の下にもぐっています。ボールが地面よりも上で揺れるようにする必要があります。

オブジェクトレベルに戻ります。***soccerball_ctrl*** Null オブジェクトを選択します。**Translate Y** を **RMB クリック**し、**Motion FX > Limit** を選択します。**Minimum** を **0** に、**Maximum** を **6** に設定します。これで、ボールはずっと上下に揺れ動くのではなく、いくらか揺れながら地上を移動するようになります。



11 ボールが弾んでいる間は、ノイズは必要ありません。ノイズが必要なのは、ボールが転がりはじめてからです。**Amplitude** のキーフレームを設定して、ノイズをオン/オフできるようにします。

新しく作成した ***motionfx*** ネットワークで、***noise1*** CHOP ノードを選択します。ボールが弾まなくなり、転がり始める**フレーム 37** に移動します。**Amplitude** を **Alt クリック**して、キーフレームを設定します。**フレーム 1** に移動して、**Amplitude** を **0** に設定します。**Amplitude** を再度 **Alt クリック**して、2 つ目のキーフレームを設定します。**Animation Editor** でカーブを選択します。Function ツールバーの **Constant** ボタンをクリックします。これで、振幅なしから振幅 1 へとシャープに切り替わります。



12 完了したら、ウィンドウを閉じます。**Motion Path** ハンドルを **RMB クリック**して **Persistent** をオフにし、選択解除したらハンドルが表示されないようにします。

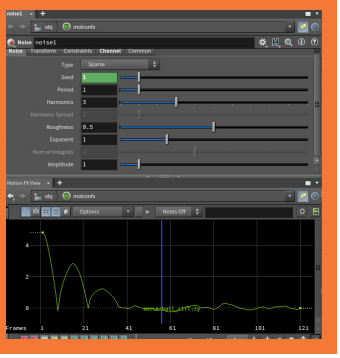
Scene View の左側にあるツールバーで、**Render Flipbook** ボタンをクリックします。デフォルト設定のまま、**Start** をクリックします。シーケンスがキャプチャされるまで待つと、**Mplay** ウィンドウにフリップブックが表示されます。これを**再生**したりスクラブして、動きを評価します。

作業内容を**保存**します。

Motion FX

キーフレームとアニメーションカーブはノードのパラメータに格納されますが、**チャンネルオペレータ（CHOP）**を使用すると、よりプロシージャルなノードベースのアプローチでモーションを操作することができます。

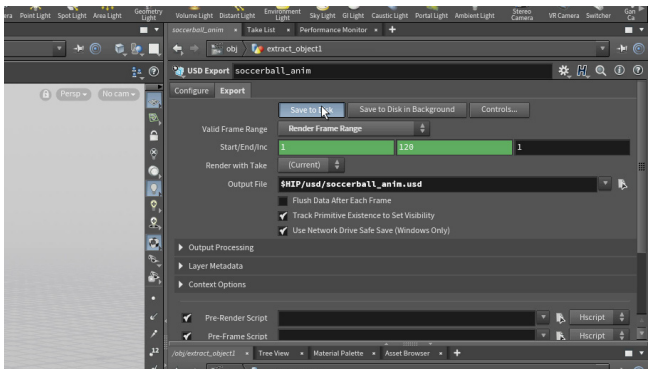
Motion FX は、**Channel CHOP** に抽出および格納される、キーフレームによる動きに適用できます。その後、**Cycle**、**Noise**、**Smooth**、**Limit**、**Lag** などのエフェクトを既存の動きに適用できます。**Constraints** シェルフにはさまざまなツールがあり、パラメータ設定によってターゲットの方を**向くようにしたり**、**遅延させたり**、**微震する**ようにできます。



パート 9

ライト、カメラ、アクション!

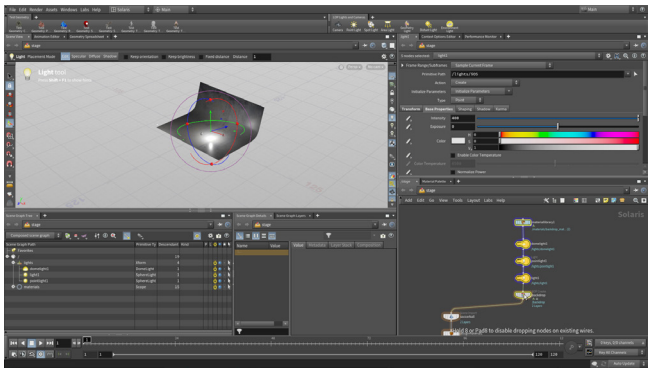
アニメートしたサッカーボールをレンダリングするには、Solaris 環境に戻って 2 つ目のショットをセットアップする必要があります。まず、背景ジオメトリから新しい LOP ノードを分岐させてから、バウンドするサッカーボールのアニメーションに合わせてライトとカメラを調整します。また、変形するジオメトリに対してモーションブラーもセットアップします。



01

オブジェクトレベルに移動します。***soccerball_anim*** を**ダブルクリック**して中に入ります。**N を押し**、すべてのジオメトリを選択したら、**Modify** シェルフで **Extract** を選択します。これにより、ボールのすべての動きと曲げが 1 つのネットワークにまとめられます。

ボールを ***extract_object*** という新しいオブジェクトに抽出している ***objectmerge*** ノードが表示されます。その出力を **RMB クリック**し、**USD Export** を見つけたら、クリックしてこのノードを配置します。このノードを ***soccerball_anim*** という**名前に変更**します。**Export** タブをクリックし、**Valid Frame Range** を **Render Frame Range** に、**Output File** を ***$HIP/geo/soccerball_anim.usd*** に設定します。**Save to Disk** ボタンをクリックします。

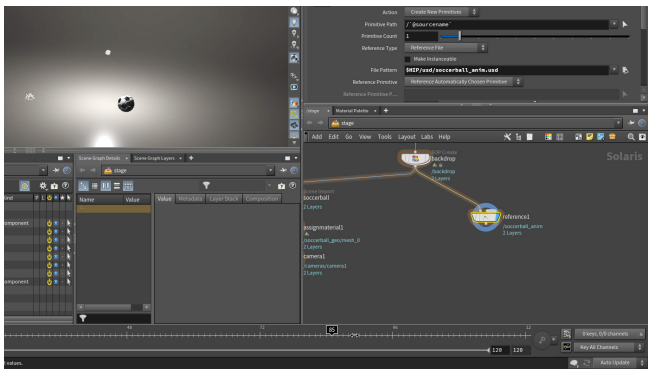


02

Solaris デスクトップに戻り、***/stage*** を指定します。**ネットワークビュー**で、***karmarendersettings*** ノードの直前に **Null** ノードを追加し、その名前を ***SHOT_01*** に変更します。

3 つの **light** ノード(***lightmixer*** ではありません)を接続解除し、それらを ***backdrop*** ノードの上に移動します。これにより、最初のショットのルックは変わらず、2 つのショット間でノードを共有できるようになります。

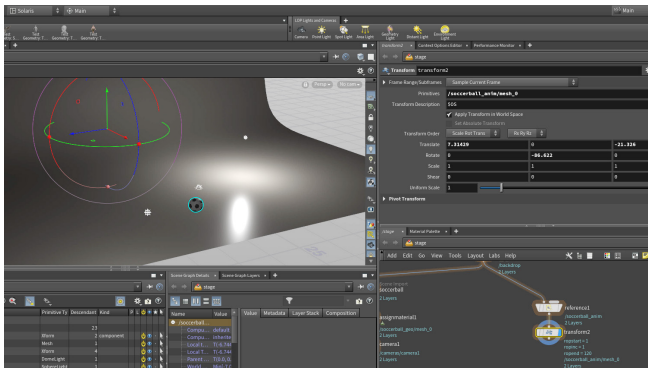
backdrop、***light***、***materiallibrary*** ノードを右に移動します。



03

ズームインして、**Reference** ノードを ***backdrop*** ノードの右下に追加します。***backdrop*** ノードをこの新しいノードに接続し、**Display フラグ**を設定します。**File Pattern** の横にある **File Chooser** をクリックし、***soccerball_anim.usd*** ファイルを見つけます。ノードの名前を ***soccerball_anim*** に変更します。

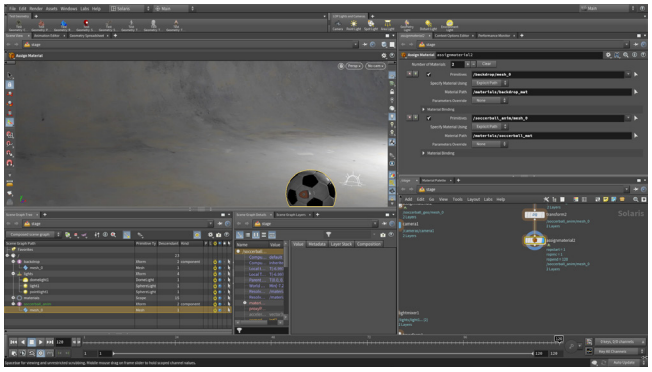
Houdini GL ビューでタイムラインをスクラブすると、USD ファイルの一部であるキャッシュ化されたアニメーションを確認できます。



04

Select ツールを使用して、新しいアニメートされたサッカーボールをクリックします。Scene View で、**Tab > Transform** を押して、Transform ノードをグラフに追加します。

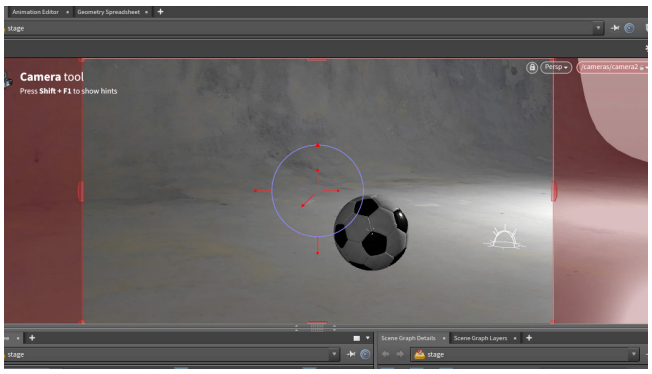
Scene View で、トランスフォームハンドルを使用してボールを中央の背景の後方に**移動**します。タイムラインを**スクラブ**して、ボールが弾みながら右に進むことを確認します。フレーム 80 あたりで止めます。**R を押して**回転ハンドルを使用できるようにします。ボールを**回転**し、背景から斜めにバウンドしてくるように移動します。タイムラインを**スクラブ**して、方向に問題がないことを確認します。



05

ネットワークビューで、SHOT 1 ネットワークから ***assignmaterial*** ノードを**選択**し、**Alt ドラッグ**してこのノードのコピーを作成します。***transform*** ノードを ***assignmaterial*** ノードに接続したら、**Display フラグ**を設定します。これによりマテリアルが***背景***に割り当てられますが、サッカーボールのプリミティブが変更されているため、マテリアルを割り当て直す必要があります。

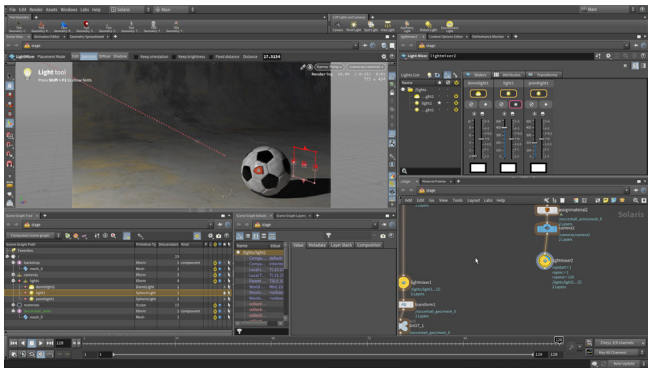
soccerball_mat の **Primitives** の横にあるフィールドで、プリミティブの名前を ***/soccerball_anim*** に変更し、新しいジオメトリにマテリアルを割り当て直します。



06

左上から右下へと、ボールがカメラに向かってアニメートされる様子が見えるまで、タンブルします。**LOP Lights and Cameras** シェルフで、Camera ツールを **Alt クリック**して、現在見ているアングルにカメラを配置します。

Lock Camera/Light to View ボタンを押し、ビュー変更に応じてカメラの位置が更新されるようにします。次にビューポートで**タンブル**、**パン**、**ドリー**してカメラを微調整し、ショットに適したフレーミングにします。タイムラインをスクラブし、シーケンス全体でカメラが機能していることを確認します。



07

camera の後に **Light Mixer** ノードを追加します。***lightmixer*** ノードでは、ライト上にカーソルを移動します。こうすると、前に紹介したライトハンドルを使用して、このショットのライティングを決定したり、ビューポートの **Karma** 表示を使ってセットアップを確認することができます。***lightmixer*** ノードを使用すると、このショットの**強度**や**露出**をさまざまに変えて試すことも可能です。

これらの編集は ***lightmixer*** ノードに保持され、元のライトは変更されません。***lightmixer*** では、マルチショットセットアップを使用しながら、既存のライトを微調整できます。

08

ネットワークビューで、***SHOT_01***、***karmarendersettings***、***usdrender_rop*** ノードを **Alt ドラッグ**します。***lightmixer*** ノードをこのチェーンに接続します。新しい ***karmarendersettings*** ノードを選択し、**Camera** が ***/camera2*** に設定されていることを確認します。**Valid Frame Range** を **Render Frame Range** に、**Output Picture** を ***$HIP/render/anim/soccerball_anim_$F2.exr*** に設定します。***$F2*** によって、レンダリングにフレーム番号と 2 のパディングが追加され、***/anim/*** によって、これらのフレームを保持するためのディレクトリが作成されます。

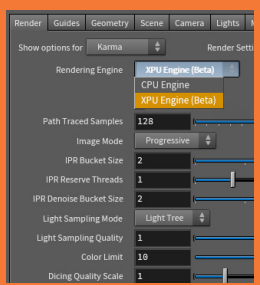
usdrender_rop ノードで、**Render to Disk** をクリックします。完了したら、**Render > Mplay > Load Disk Files** を選択し、レンダリングした画像を開いて最終的なシーケンスを確認します。作業内容を**保存**します。

KARMA レンダラ

Karma は、Solaris/LOP コンテキストで **USD** ファイルを使用できるよう設計された物理ベースの **HYDRA** レンダラです。**ビューポート**で使用してインタラクティブに更新したり、**Karma** ノードを使用してディスクにレンダリングできます。

注 : Houdini 19 は、**Karma XPU** レンダリングエンジンのプレビューに対応しています。このハイブリッドな GPU/CPU レンダラは**アルファ**品質で、多くの機能はまだ開発中であるため、**テスト目的でのみ**使用します。**XPU** は、Scene View の **Display Options** または **Karma** ノードで選択できます。

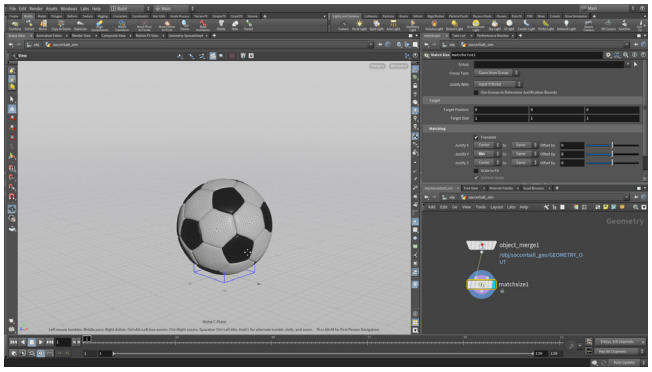
ビューポートメニュー



Display Options

パート 10
リジッドボディシミュレーションのセットアップ

1 個のサッカーボールをアニメートするには、従来のアニメーション方法が適していますが、たくさんのサッカーボールをアニメートしたい場合は、ダイナミクスの使用がお勧めです。ダイナミクスを使うには、ソルバがフレームごとに各オブジェクトの相互作用を判断できるよう、シミュレーションを行う必要があります。パックドジオメトリを使用すると、効率よくシミュレーションの結果を得られます。

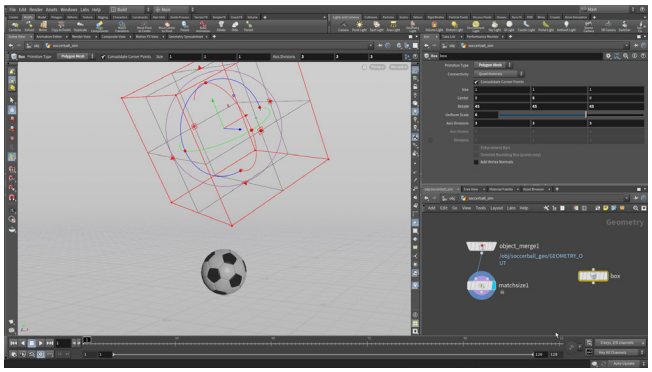


01

Build デスクトップに戻り、オブジェクトレベルに移動します。すべてのアニメーションリグノードと ***extract_object*** ノードを、Display フラグをオフにして非表示にします。***soccerball_geo*** の表示をオンにします。

soccerball_geo ノードを選択してから、**Modify** シェルフの **Extract** ツールをクリックします。これにより、サッカーボールのオブジェクトが結合された、新しいオブジェクトが作成されます。1 つ上のレベルに移動し、***extract_object*** の名前を ***soccerball_sim*** に変更します。***soccerball_geo*** オブジェクトを非表示にします。

soccerball_sim オブジェクトの中に戻り、ジオメトリを作業していきます。**Match Size** ノードを追加して、原点を基準とした中央にボールを配置します。



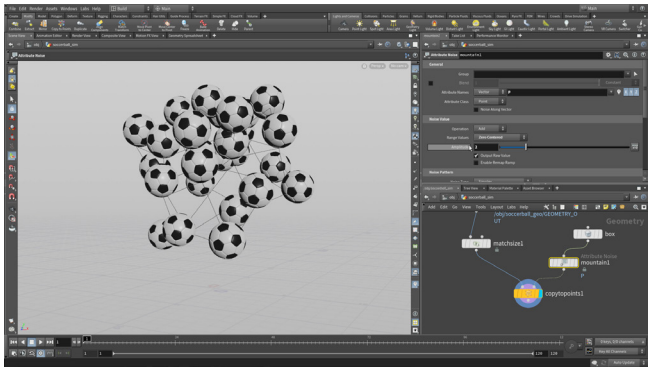
02

ネットワークビューで、**Tab > Box** を押し、それを ***matchsize*** ノードの右側に配置します。

box ノードで次のように設定します。

- **Center** を **0, 8, 0** にする
- **Rotate** を **45, 45, 45** にする
- **Primitive Type** を **Polygon Mesh** にする
- **Uniform Scale** を **6** にする
- **Axis Divisions** を **3, 3, 3** にする

これで、シミュレーションでの位置が適切になります。

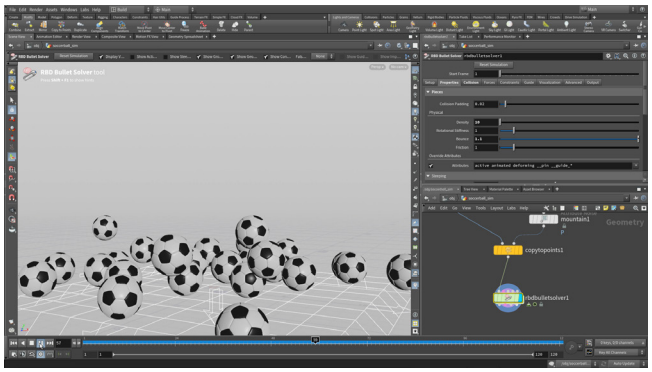


03

ネットワークビューで、その他のノードのすぐ下に **Copy to Points** ノードを追加します。***matchsize*** ノードを 1 つ目の入力に接続し、***box*** ノードを 2 つ目の入力に接続します。

Pack and Instance オプションを**オン**にします。これにより、ジオメトリが立方体のポイントにインスタンス化されるので、シミュレーションが高速化します。***copytopoints*** ノードに **Display フラグ**を設定します。

ネットワークビューで、**Tab > Mountain** を押して、***box*** と ***copytopoint*** ノードの間にそのノードを配置します。**Noise Along Vector** オプションを**オフ**にしてから、**Amplitude** を **2** に、**Range Values** を **Zero Centered** に設定します。これで、ボックスのポイントが微震するようになります。



04

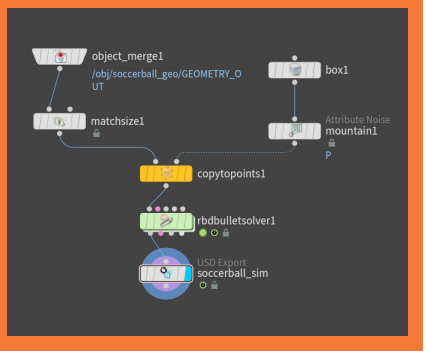
フレーム 1 にいることを確認します。***copytopoints*** ノードの後に **RBD Bullet Solver** ノードを追加します。**Collision タブ**をクリックし、**Ground Collision** までスクロールダウンして、**Ground Type** を **Ground Plane** に設定します。**Play** を押して、シミュレーションをテストします。シミュレーションはキャッシュ化され、タイムラインをスクラブして結果を確認できるようになります。

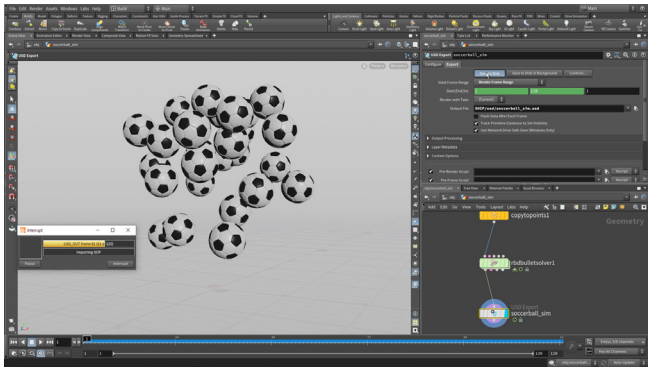
Collisions タブで、**Bounce** を **0.8** に設定します。**Properties** タブで、**Density** を **10** に、**Bounce** を **1.1** に設定します。このノードのパラメータエディタの上部で、**Reset Simulation** ボタンをクリックしてから **Play** を押し、再度シミュレーションを実行します。スクラブして確認します。

SOP 内に隠れた DOP

Houdini では、シミュレーションは **Dynamic Operators**、つまり **DOP** を使用して処理されます。**Geometry/SOP** コンテキストの **RBD Bullet Solver** ノードでは、内部にダイナミクスネットワークが埋め込まれたノードを使用することになります。

これにより、すべての DOP ノードを接続して準備を整え、かつビューに表示されてない状態を簡単にジオメトリレベルでセットアップできるようになりました。シンプルなセットアップでは、ジオメトリレベルで作業することで適切なシミュレーションを得られます。さまざまなソルバをより詳細に制御する必要がある場合は、DOP で直接作業する必要があります。



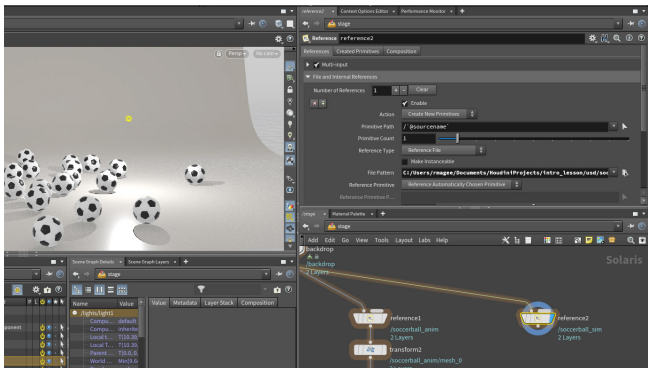


05

チェーンの終端に **USD Export** ノードを追加し、**Display フラグ**を設定したら、それを ***soccerball_sim*** という**名前に変更**します。

Valid Frame Range を **Render Frame Range** に、**Output File** を ***$HIP/geo/soccerball_sim.usd*** に設定します。

Save to Disk ボタンをクリックして、USD ファイルを geo ディレクトリに保存します。このキャッシュ化されたアセットは、3 つ目のショットとして Solaris セットアップで参照されます。

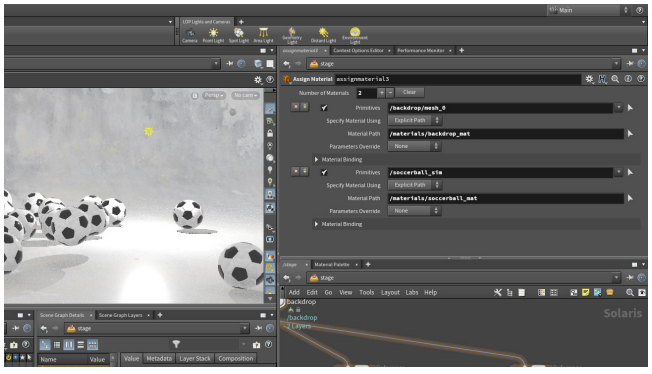


06

デスクトップを **Solaris** に戻し、パスを ***/stage*** に設定します。***Persp*** メニューで **Houdini GL** が選択されていることを確認します。

soccerball_anim **Reference** ノードを Alt ドラグし、その **Display フラグ**を設定します。**File Pattern** を ***$HIP/geo/soccerball_sim.usd*** に設定します。

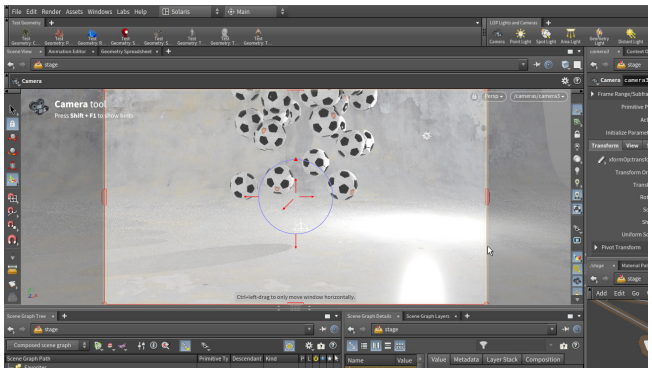
このノードを ***soccerball_anim*** という**名前に変更**します。



07

ネットワークビューで、SHOT 2 ネットワークから ***assignmaterial*** ノードを**選択**し、**Alt ドラッグ**してこのノードのコピーを作成します。***soccerball_anim*** ノードを ***assignmaterial*** ノードに接続したら、その **Display フラグ**を設定します。これによりマテリアルが ***背景***に割り当てられますが、サッカーボールのプリミティブが変更されているため、マテリアルを割り当て直す必要があります。

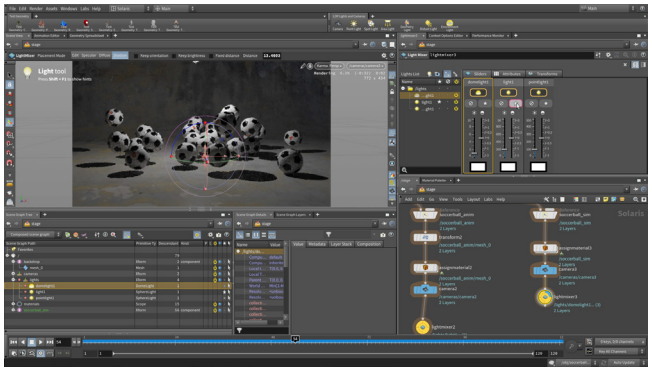
soccerball_mat の ***Primitives*** の横にあるフィールドで、プリミティブの名前を ***/soccerball_sim*** に変更し、新しいジオメトリにマテリアルを割り当て直します。



08

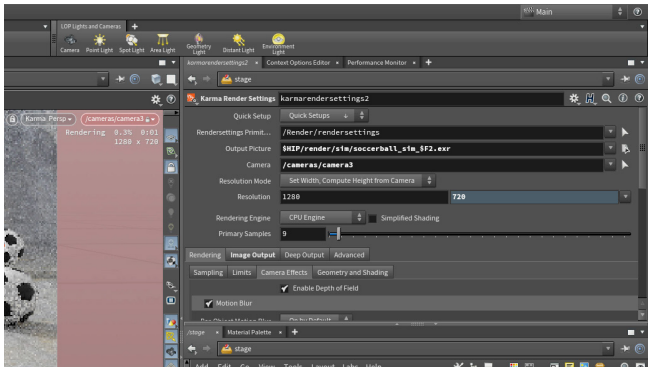
ボールがカメラに向かってアニメートされる様子が見えるまで、タンブルします。**LOP Lights and Cameras** シェルフで、Camera ツールを **Alt クリック**して、現在見ているアングルにカメラを配置します。

Lock Camera/Light to View ボタンを押し、ビュー変更に応じてカメラの位置が更新されるようにします。次にビューポートで**タンブル**、**パン**、**ドリー**してカメラを微調整し、ショットに適したフレーミングにします。タイムラインをスクラブし、シーケンス全体でカメラが機能していることを確認します。

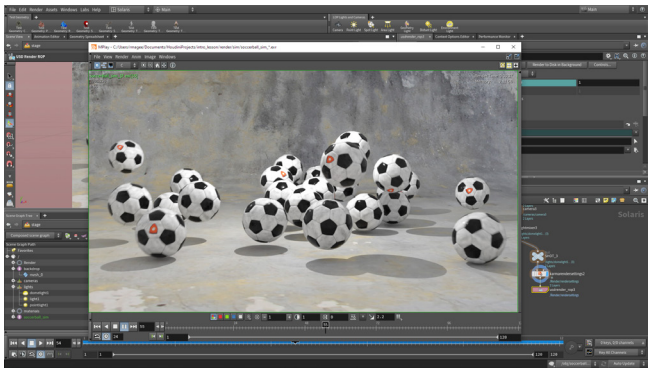


09 camera の後に **Light Mixer** ノードを追加します。***lightmixer*** ノードでは、ライト上にカーソルを移動します。こうすると、前に紹介したライトハンドルを使用して、このショットのライティングを決定したり、ビューポートの **Karma** 表示を使ってセットアップを確認することができます。***lightmixer*** ノードを使用すると、このショットの**強度**や**露出**をさまざまに変えて試すことも可能です。

これらの編集は ***lightmixer*** ノードに保持され、元のライトは変更されません。***lightmixer*** では、マルチショットセットアップを使用しながら、既存のライトを微調整できます。



10 ネットワークビューで、***SHOT_02***、***karmarendersettings***、***usdrender_rop*** ノードを **Alt ドラッグ**します。新しい ***lightmixer*** ノードをこのチェーンに接続します。新しい ***karmarendersettings*** ノードを選択し、**Camera** が ***/camera3*** に設定されていることを確認します。



11 usdrender_rop ノードで、**Valid Frame Range** を **Render Frame Range** に、**Output Picture** を ***$HIP/render/sim/soccerball_sim_$F2.exr*** に設定します。**Render to Disk** をクリックします。

完了したら、**Render > Mplay > Load Disk Files** を選択し、レンダリングした画像を開いて最終的なシーケンスを確認します。

まとめ

Houdini のさまざまな側面に触れながら、ゼロからシーンを構築してきました。モデリング、テクスチャのセットアップ、レンダリング、シミュレーションを行いました。その過程で、さまざまな Houdini コンテキストと、それらの切り替え方法についても学習しました。

このレッスンは、VFX の大作を作るものではありません。Houdini を深く探り、その包括的なツールセットを探求するにあたって必要となる基本スキルを紹介するものです。

SideFX の Web サイトには、スキルアップのための学習教材が豊富に用意されています。

最高の旅になりますように！

HOUDINI FUNDAMENTALS

ノード、ネットワーク、デジタルアセット

Houdini のノードベースワークフローを理解するには、実際のプロジェクトで使ってみるのが一番です。重要なのは、プロシージャルな考え方や作業方法を学ぶことです。このレッスンでは、プロシージャルなノードとネットワークを使用してカスタムの **brickify**（ブロック化）ツールを自作し、機能やインターフェースを定義する方法について学習します。

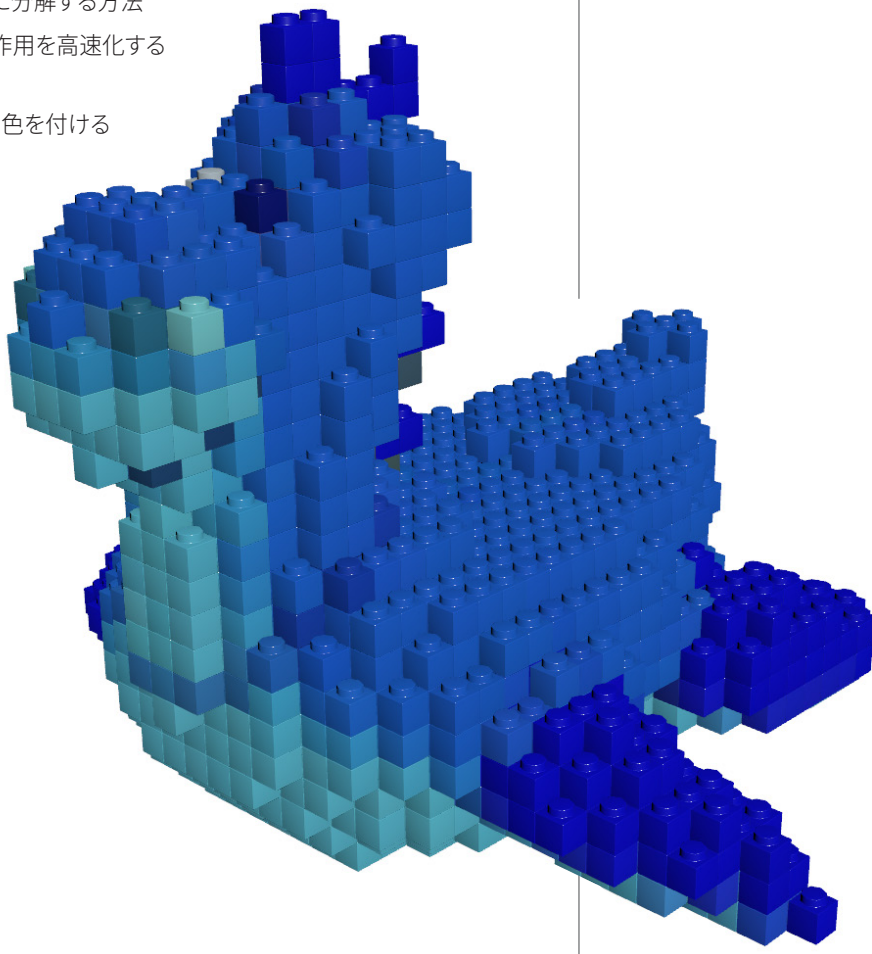
その過程で、Houdini ワークスペースのさまざまな機能を使います。冒頭の概要を参照して、UI 要素の仕組みを確認しておきましょう。このレッスンは、アイデアを実践に移す場です。実践は、最も効果的な学習方法の 1 つです。

レッスンの目標

- ***任意の 3D 形状をおもちゃのブロックに変えるカスタムツールを作成します。***

学習内容

- プラスチックの連結ブロックをモデリングする方法
- デフォルトのゴムのおもちゃの形状をグリッドポイントに分解する方法
- パックプリミティブとインスタンス化を活用して、相互作用を高速化する方法
- アトリビュートを使って、テクスチャマップでブロックに色を付ける方法
- ノードとネットワークを使用してデータの流れを制御する方法
- デジタルアセットを作成して、ソリューションをパッケージ化したり、他の人と共有する方法
- 徐々に現れるブロックをアニメートする方法



使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、以下の Houdini 製品で実行可能です。

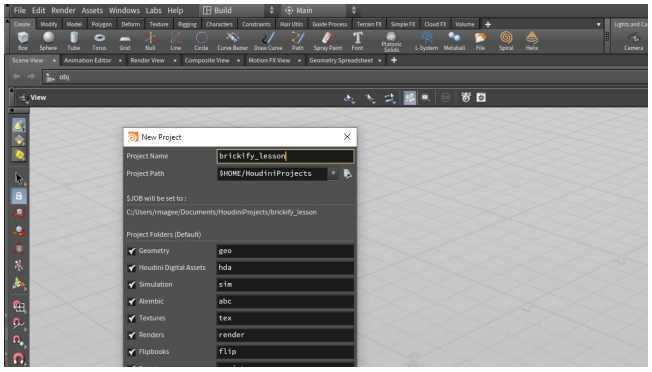
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

ドキュメントバージョン 4.0.1J | 2023 年 8 月

パート 1

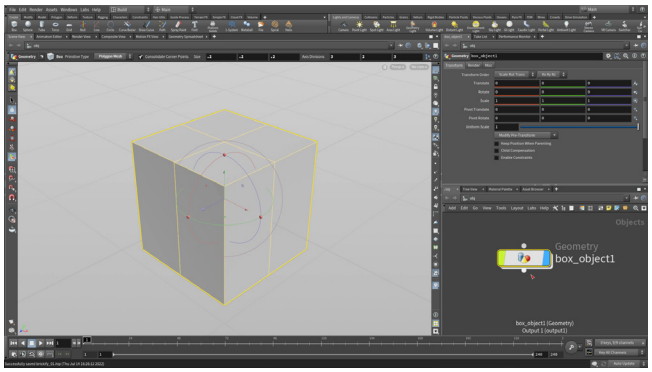
シングルブロックの作成

最初はシングルブロックモデルを作成します。その後、モデルをポイントにコピーして、ブロック化した形状を作成します。この形状を作成するために、いくつかのポリゴンモデリングツールを使用します。その際、Houdini でアクションを実行するたびにノードが作成されますが、それはジオメトリ作成手順のレシピとなります。



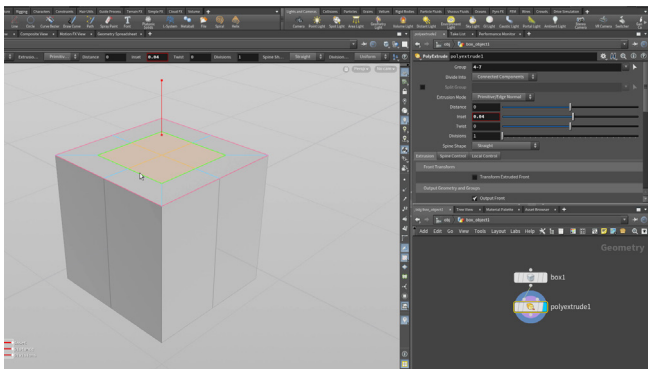
01 **File > New Project** を選択します。**Project Name** を ***brickify_lesson*** に変更し、**Accept** を押します。これにより、プロジェクトディレクトリとサブフォルダが作成され、このショットに関連するすべてのファイルがそこに配置されます。

File > Save As... を選択します。新しい ***brickify_lesson*** ディレクトリが表示されているはずです。ファイル名を ***bricks_01.hip*** に設定し、**Accept** をクリックして保存します。



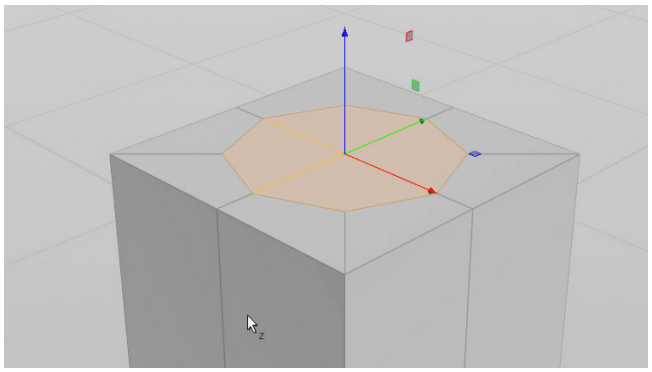
02 ビューポートで、**C を押して** Radial メニューを表示します。メニューから **Create > Geometry > Box** を選択します。カーソルの位置に、シーン内への配置待ちの状態にあるボックスの輪郭が表示されます。**Enter を押して**、原点の位置に配置します。**オペレーションコントロール**ツールバーで、**Size** を **0.2, 0.2, 0.2**、**Axis Divisions** を **3, 2, 3** に設定します。

ネットワークビューに ***box_object*** が表示されています。このオブジェクトレベルのノードには、この形状のトランスフォーム情報が含まれています。**オペレーションコントロール**ツールバーには、1 つ下のレベルの別の ***box*** ノードのパラメータが表示されています。



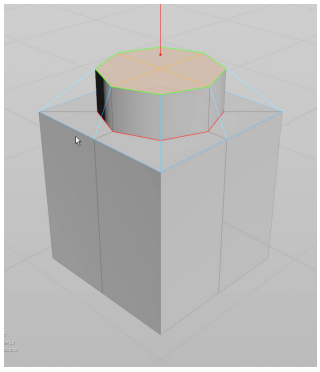
03 **Select** ツールをクリックし、**4 を押して**プリミティブ選択モードに切り替えます。ボックス上部の 4 つのフェースを選択します。**C を押して** Radial メニューを表示し、**Model > Polygons > PolyExtrude** を選択します。ネットワークビューで、***box*** ノードが ***polyextrude*** ノードに接続されているのを確認できます。

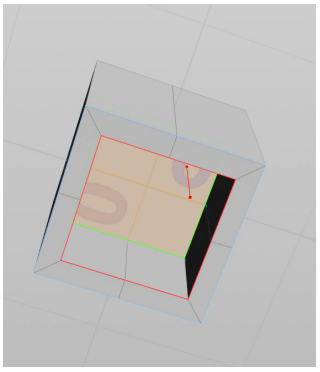
パラメータエディタで、スライダを使用して **Inset** を **0.04** に設定すると、ボックスの上面に新しいポリゴンが作成されます。各ノードには、そのノードの目的に関連するパラメータが含まれています。これらはジオメトリノードで、SOP (Surface Operator) とも呼ばれています。



04 次に、**T** キーを押して Move ツールを呼び出します。これにより、ネットワークに ***Edit SOP*** ノードが追加されます。ビューポートの何もない空間で **RMB クリック**してメニューを表示し、**Make Circle** を選択して、選択したポリゴンに丸みを付けます。

このメニューは ***Edit*** ノードに関連付けられています。すべてのノードに独自のインターフェースがあり、関連するツールがアクティブになっている場合のみアクセス可能です。ここでは、Move ツールでハンドルにアクセスしています。**Handle** ツールを使用した場合は、ノードのインタラクティブなハンドルにアクセスできます。



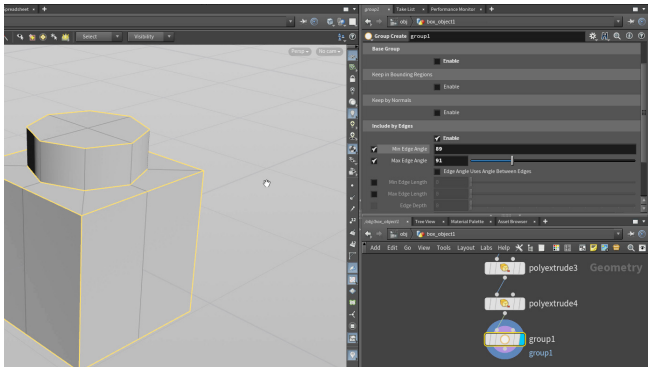


05

C を押して Radial メニューを表示し、**Model > Polygons > PolyExtrude** を選択します。Scene View ペインで、ハンドルを使用してポリゴンを上にドラッグし、**Distance** を **0.05** に設定します。

タンブルしたら、**S を押して**選択モードに切り替え、ボックス底部のポリゴン 4 つを選択します。**Q を押して PolyExtrude** ツールを繰り返します。パラメータエディタで、スライダを使用して **Inset** を **0.025** に設定します。

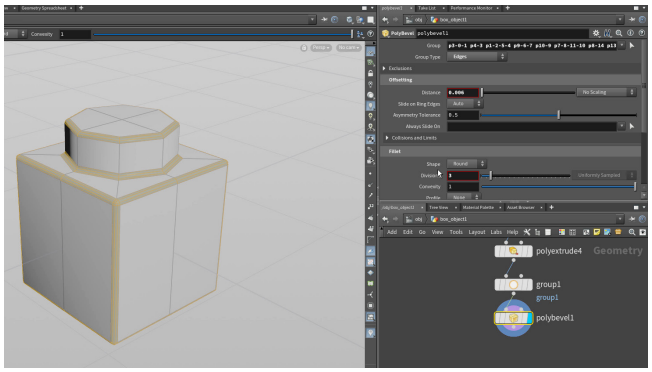
Q を押してツールを繰り返し、**Distance** を **-0.175** に設定します。完了したら、再度タンブルしてブロックの上面が見えるようにします。



06

3 を押してエッジ選択に切り替え、N を押してすべてのエッジを選択します。Scene View で Tab を押し、**Group...** と入力していき、**Group** を選択します。パラメータエディタで、**Group Name** を ***bevel_edges*** に設定します。

次に、**Base Group** の **Enable** を**オフ**にし、**Include by Edges** セクションの Enable をオンに設定します。**Min Edge Angle** をオンにして **89** に設定したら、**Max Edge Angle** をオンにして **91** に設定します。

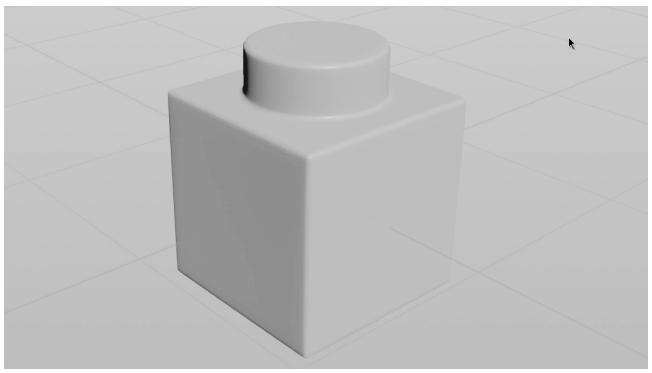


07

S を押して Select ツールに切り替えます。**9 を押して「Select Groups」**オプションをオンにします。ポップアップウィンドウで、***bevel_edges*** グループをクリックします。

ビューポートで、**C を押して** Radial メニューを表示します。メニューから **Model > Polygons > PolyBevel** を選択します。polybevel ノードが追加され、**Group** フィールドには ***bevel_edges*** が自動的に入力されます。

Bevel Offset を **0.006** に設定します。**Fillet** で、**Shape** を **Round**、**Divisions** を **3** に設定します。



08

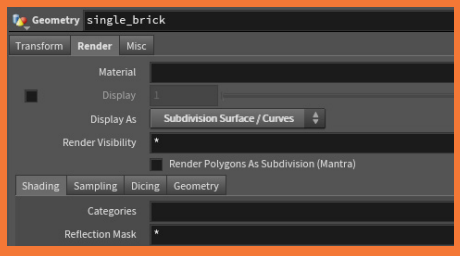
オブジェクトレベルに移動し、ネットワークビューでオブジェクトの名前を ***single_brick*** に変更します。ブロックを選択した状態で、**Shift +** を押して、この形状のサブディビジョンサーフェスの表示をオンにします。ブロックを選択解除して、細分化されたモデルを表示します。オブジェクトにワイヤーラインが見えたら、**V を押し**、Radial メニューから **Shading > Smooth Shading** を選択して非表示にします。

作業内容を**保存**します。

サブディビジョンの表示

Shift + および **Shift -** を使用すると、選択したポリゴンオブジェクトのサブディビジョン表示をオンまたはオフにできます。オンにすると、ビューポートで細分化が行われ、細分化された形状のルックを確認できます。これらのホットキーは、**Display As** パラメータを設定するものです。このパラメータは、オブジェクトレベルのオブジェクトの **Render** タブにあります。

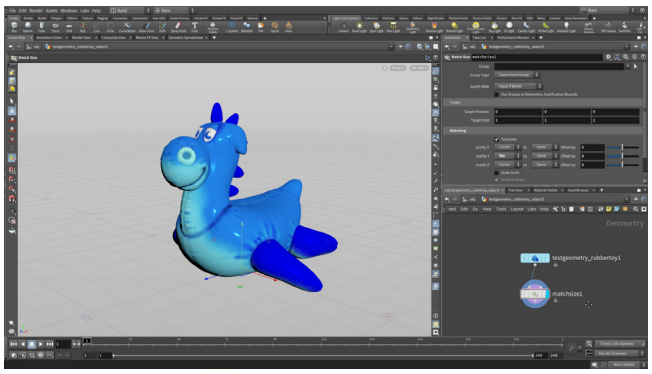
オブジェクトをサブディビジョンとしてレンダリングするには、同じタブで **Render Polygons As Subdivision (Mantra)** をオンにする必要があります。



パート 2

ポイントクラウドにブロックをコピー

ここでは、特定のジオメトリの形状と一致するポイントクラウドを作成します。その後、ブロックを 3D グリッドにインスタンス化して、ブロック化したバージョンを作成します。インスタンスは、ブロックジオメトリをパック化して、それらをポイントにインスタンス化することで生成できます。

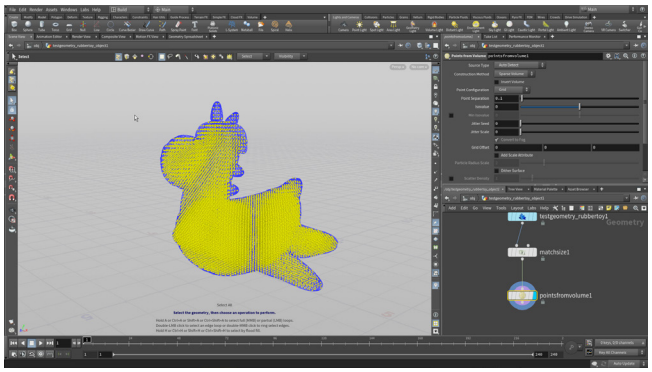


01 Scene View で、Tab を押して **Test...** と入力していきます。**Test Geometry: Rubber Toy** を選択します。**Enter** を押して、原点に配置します。

I を押して、***test geometry*** オブジェクトの中に入ります。次のように設定します。

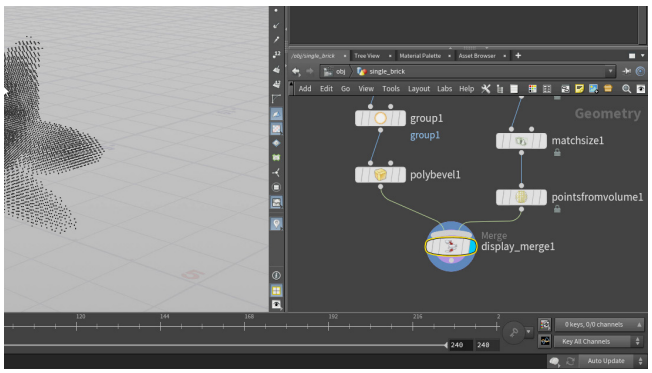
- **Uniform Scale** を **3** にする

Match Size ノードを追加して、**Justify Y** を **Min** に設定します。おもちゃが持ち上がり、地面の上に配置されます。



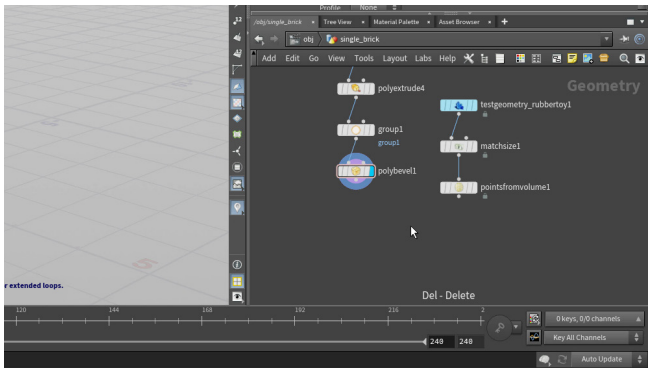
02 ネットワークエディタで、***matchsize*** ノードの出力を **RMB クリック**し、**Points...** と入力して **Points from Volumes** を選択し、そのノードをネットワークに配置します。次に **Display フラグ**を設定して、この新しいノードの出力に集中できるようにします。

S を押して Select ツールに切り替え、**2 を押して**ポイント選択にします。**N を押して**すべてのポイントを選択すると、それらが黄色でハイライトされます。これらのポイントにブロックをコピーしていきます。

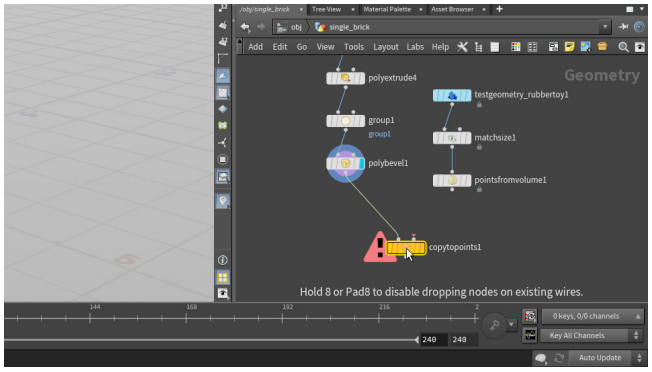


03 **U を押して**オブジェクトレベルに戻り、ゴムのおもちゃの名前を短く ***rubbertoy*** にします。ネットワークビューで **Shift** キーを押し、***rubbertoy*** をクリックしてから ***single_brick*** をクリックします。

Modify シェルフタブで **Combine** を選択し、これらのオブジェクトを結合します。ジオメトリレベルでは、ノードが merge ノードに接続されているのが分かります。



04 ネットワークビューで、***display_merge*** ノードを選択し、**Delete** キーを押します。これで、***polybevel*** ノードチェーンが表示され、他のチェーンは非表示になります。Houdini では、オブジェクトレベルに戻るときに、どのノードを表示するのかを選択できます。

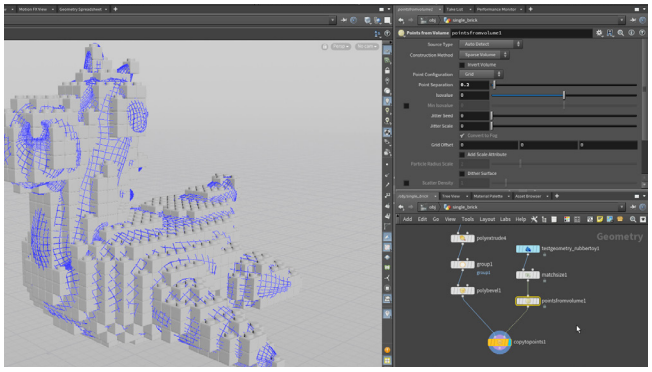


05

polybevel ノードの出力を **RMB クリック**して、**copy...** と入力していき、**Copy to Points** を選択します。そのノードをクリックして 2 つのチェーンの下に配置し、**Display フラグ**を設定します。

Pack and Instance オプションをオンにします。このオプションをオンにすると、オフのときよりもずっと速くコピーしたブロックを表示できます。

この時点でノードにエラーが表示されますが、それは 2 つ目の入力を接続していないからです。

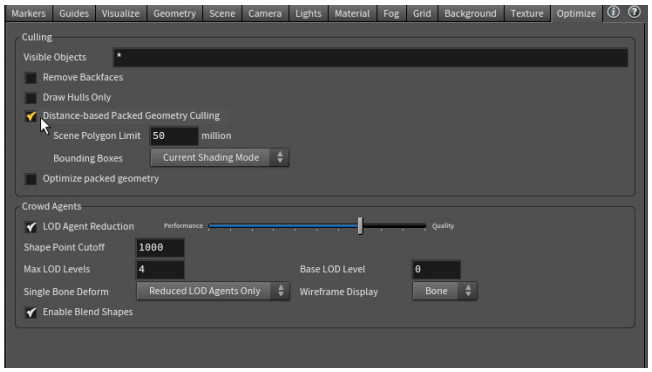


06

pointsfromvolume ノードの下のドットをクリックして、***copytopoints*** ノードの 2 つ目の入力に接続します。

ブロックが重なっているように見えます。***pointsfromvolume*** モードに戻り、**Point Separation** を **0.2** に設定します。

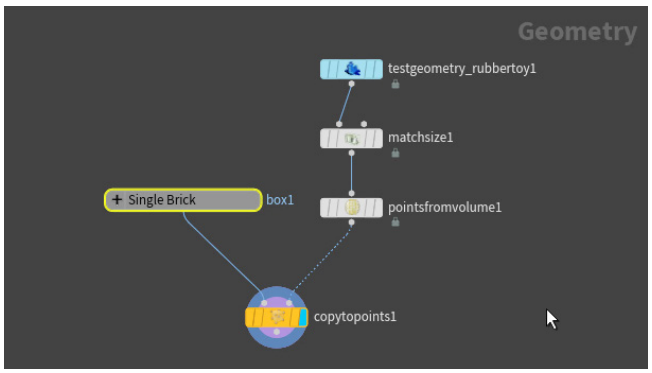
次に、ブロックをグリッドポイントにコピーします。



07

copytopoints ノードをクリックします。ブロックの一部がダークグレーで表示され、適切なブロックを表示していません。ビューポートでのジオメトリカリングが原因です。これは、表示設定で修正できます。

Scene View で、**スペースバー + D** を押して **Display Options** を表示します。**Optimize** タブをクリックし、**Scene Polygon Limit** を **50 million** よりも大きい値に設定するか、**Distance-based Packed Geometry Culling** を**オフ**にします。フローティングパネルを閉じます。これで、すべてのブロックがパックインスタンスとしてポイントにコピーされていることが確認できます。



08

作業を保存する前に、ネットワークを整理しましょう。シングルブロックを構成しているノードを選択し、**Shift + O** を押してその周りにネットワークボックスを作成します。ボックスのタイトルバーをクリックし、***Single Brick*** と入力します。その後ボックスを折り畳み、下に移動してネットワークを少し整理します。

作業内容を**保存**します。

パックインスタンス

Copy to Points ノードで **Pack and Instance** オプションを**オフ**のままにすると、100 万以上のポイントとプリミティブを含む大規模なモデルになります。インスタンス化を使用しないと、ビューポートでの操作が遅くなります。

このオプションを**オン**にすると、ブロックモデルの **338 ポイント**がパック化およびインスタンス化されるため、Copy to Points ノードのポイント数を大幅に減らして効率化することができます。

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 | オフ

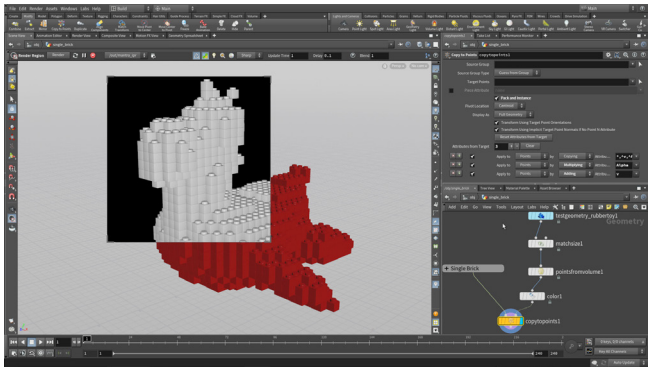
Points	3,032	Center	0, 2.025, 0
Primitives	3,032	Min	-3.1, -0.5, -2.7
Vertices	3,032	Max	3.1, 4.55, 2.7
Packed Geos	3,032	Size	6.2, 5.05, 5.4

Pack and Instance | オン

パート 3

カラーの追加とティーポットへの切り替え

ポイントにカラーを追加し、そのカラーをインスタンス化されたブロックが取得するようにします。最初、このカラー転送はビューポート内のみで適用されますが、適切なマテリアルをセットアップすると、ポイントカラーを使用してブロックをレンダリングできるようになります。その後、ゴムのおもちゃとティーポットの切り替えをセットアップし、異なる形状でもネットワークが機能することを確認します。

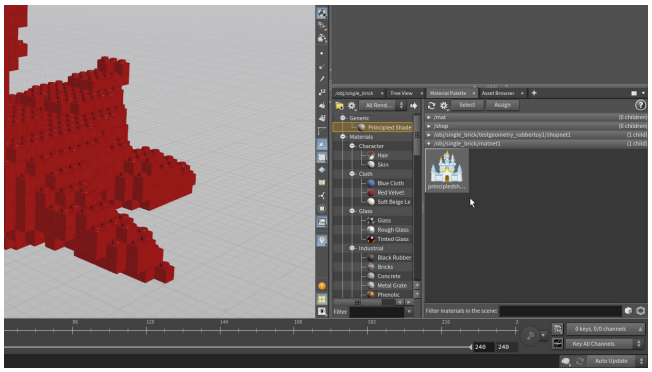


01

pointsfromvolume ノードと ***copytopoints*** ノードの間に color ノードを追加します。これによりポイントにカラーが追加され、ブロックにコピーされます。**color** を**赤色**に設定し、背景に対してブロックが目立つようにします。

Render Region ツールをクリックし、境界ボックスをゴムのおもちゃ上にドラッグします。テストレンダリングが始まり、ブロックが赤色でレンダリングされていないことが分かります。マテリアルを割り当て、カラーを取得する必要があります。

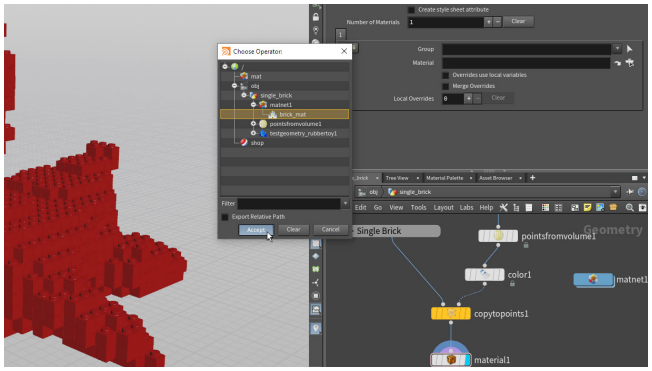
レンダリング領域の右上にある **X** ボタンをクリックして終了します。



02

ネットワークビューで **Tab** を押し、**Mat...** と入力します。**Material Network** ツールを選択し、クリックしてネットワークにノードを配置します。

Material Palette に移動して ***/mat*** を閉じ、***/matnet*** を開きます。Principled Shader を ***matnet*** にドラッグします。***brick_mat*** と名前を付けます。

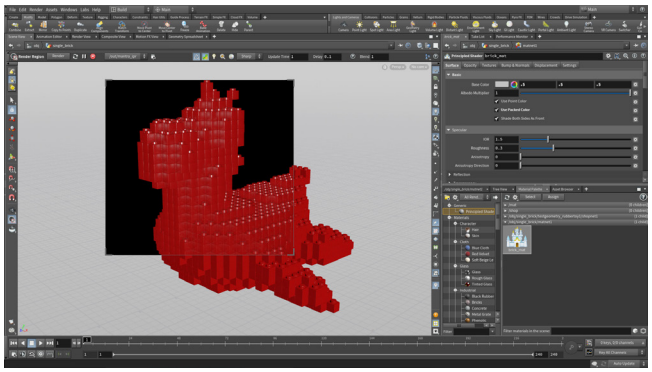


03

ネットワークビューのバックボタンを使用して、ジオメトリネットワークに戻ります。***copytopoints*** ノードの出力を **RMB クリック**して、**Material...** と入力していきます。**material** を選択し、ネットワークに配置して、**Display フラグ**を設定します。

Material パラメータの右端の **Operator Chooser** ボタンをクリックします。ポップアップウィンドウで、***brick-material*** に移動してハイライトします。**Export Relative Path オプション**をオンにして、**Accept** をクリックします。

マテリアルは割り当てられましたが、ブロックはまだレンダリングされません。



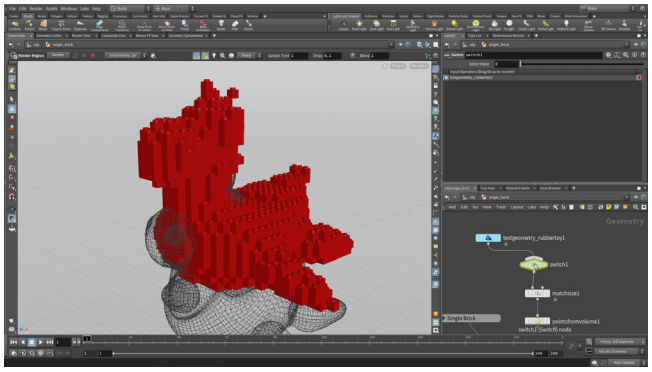
04

Material Palette に戻り、***brick_material*** を選択します。**Surface** タブをクリックして、**Base Color** を **0.5, 0.5, 0.5** に設定し、**Use Packed Color** オプションを**オン**にします。

Render Region ツールをクリックし、境界ボックスをゴムのおもちゃ上にドラッグします。レンダリングされたブロックが赤色になるはずです。

なっていない場合は、Scene View 上部のバーで **Render** をクリックして、変更が適用されていることを確認します。

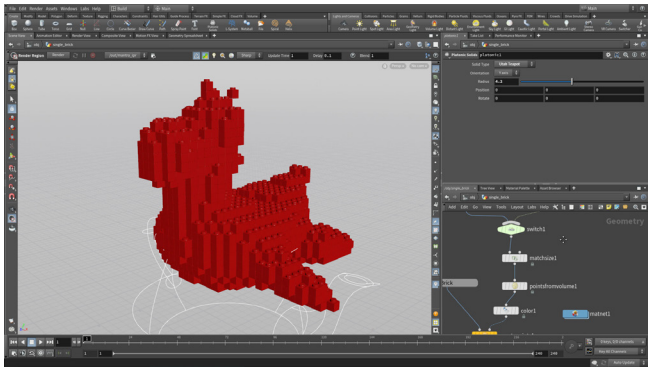
作業内容を**保存**します。レンダリング領域の右上にある **X** ボタンをクリックして終了します。



05 ネットワークビューのバックボタンを使用して、ジオメトリネットワークに戻ります。**S を押して** Select ツールに切り替えます。ネットワークビューで、**Tab** を押して **Switch** と入力し、それを **Sourcing** フォルダからネットワークビューにドラッグします。

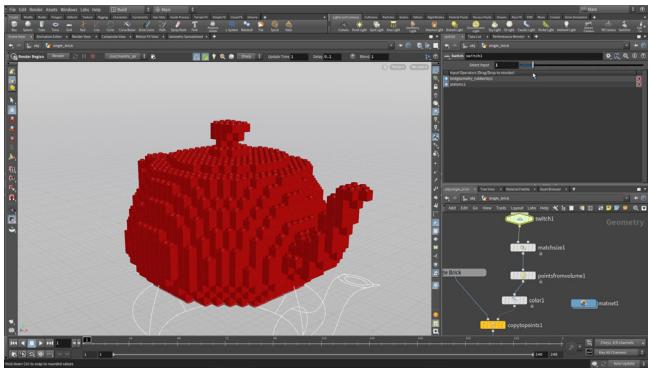
次に、***rubbertoy*** ノードと ***matchsize*** ノードを接続しているワイヤー上にドラッグします。これで、ネットワークの 2 つのノードの間に switch ノードが挿入されます。

このノードにより、異なる入力形状の間での切り替えが容易になります。



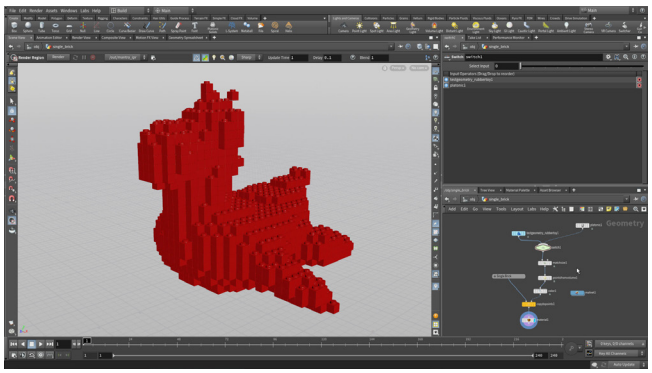
06 ツールシェルフで **Create** メニューに移動し、**Platonic** ツールをネットワークビューにドラッグします。これにより、platonic ノードがジオメトリレベルに配置されます。ツールをネットワークビューにドラッグできるのは、ノードタイプが現在のネットワークレベルに対して有効な場合のみです。

platonic ノードの **Solid Type** を **Utah Teapot** に設定します。**Radius** を **4.2** に設定します。Points from Volume ノードが新しいボリュームに合わせてポイントを更新し、異なる構成のブロックを生成します。



07 ***platonic*** ノードの出力を ***switch*** ノードの入力に接続します。***switch*** ノードを選択し、**Select Input** を **1** に変更します。ゴムのおもちゃと同じセットアップを使用して、プラトン立体が生成されます。

これは、プロシージャルシステムの最大のメリットの 1 つです。後で、このネットワークをデジタルアセットと呼ばれるカスタムツールにパッケージ化します。**デジタルアセット**なら、簡単に他の人とネットワークを共有できます。また、ツール進化に合わせて変更や更新が必要となるスタジオ環境に配備する場合でも、ツールの管理が容易です。



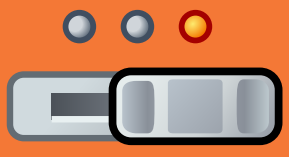
08 ***switch*** ノードで **Select Input** を **0** に設定し、***rubbertoy*** に戻ります。これら 2 つの形状を切り替えたり、さらに形状を追加してブロック化できるようになりました。

作業内容を**保存**します。

SWITCH ノード

Switch ノードは、ノードネットワークにオプションを提供する素晴らしいツールです。このノードを使用すると、さまざまなチェーンを接続 / 接続解除することなく、素早く複数のオプションを探求できます。

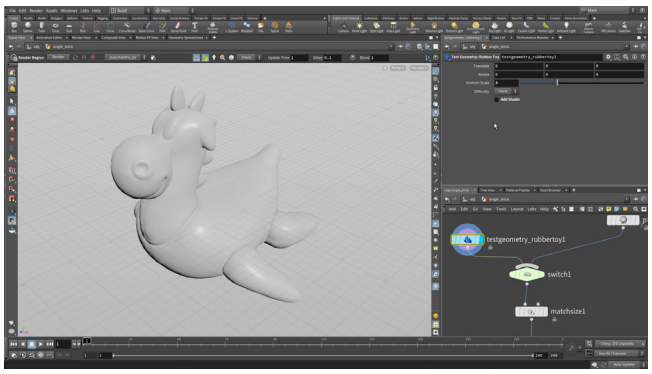
このノードはまた、ネットワークを**デジタルアセット**にまとめる際にも大変便利です。メニューまたはスライダのいずれかとしてアセットにプロモートすれば、さまざまなオプションに素早くアクセスできます。



パート 4

テクスチャを使用したポイントへのカラー付け

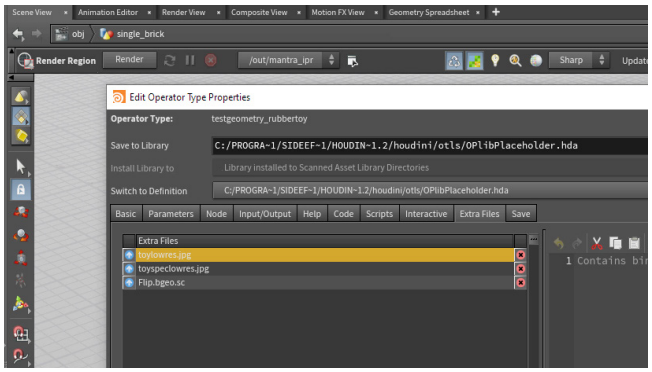
先ほどは、ポイントにカラーアトリビュートを追加して、ブロックインスタンスのカラーに影響するようにしました。
ここでは、単色の代わりにテクスチャマップを使用して、ブロックのルックをさらに面白くしていきます。
いくつか特殊なノードを使用して、テクスチャをポイントカラーに変換します。



01 ***testgeometry_rubbertoy*** の **Display フラグ**をオンにします。***rubbertoy*** ノードを選択し、パラメータエディタで **Add Shader** パラメータを**オフ**にします。

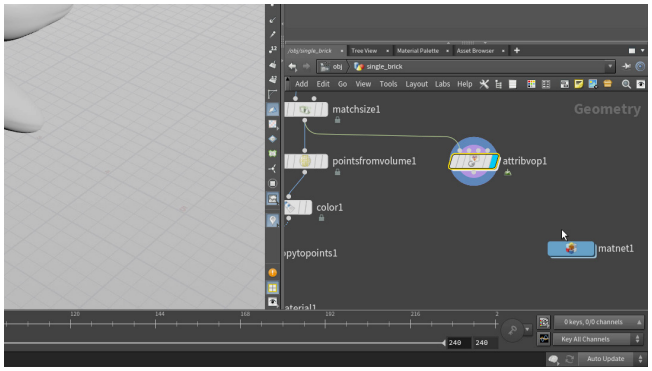
パースビューで UV を非表示にするため、**Display Options** バーに移動して **Show UV Texture when UV's Present** ボタンを**オフ**にします。

ディスク上のテクスチャマップからカラーを抽出するという別の方法で、ブロックに再度カラーを付けます。



02 **Asset** メニューから、**Edit Asset Properties > Rubber Toy** を選択します。**Type Properties** ウィンドウで、**Extra Files** タブをクリックし、***toylowres.jpg*** を選択します。

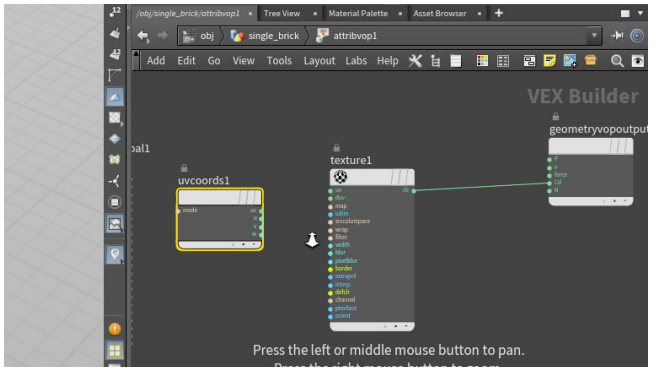
Save as File ボタンをクリックして、***tex*** フォルダに保存します。テクスチャがデジタルアセットに保存されたので、アセットと一緒に共有することができます。ディスク上のテクスチャを使用して、ブロックにカラーを追加していきます。



03 ネットワークビューで、**Tab** を押して ***Attribute...*** と入力していきます。**Attribute VOP** ノードを選択し、***pointsfromvolume*** ノードの横に配置します。

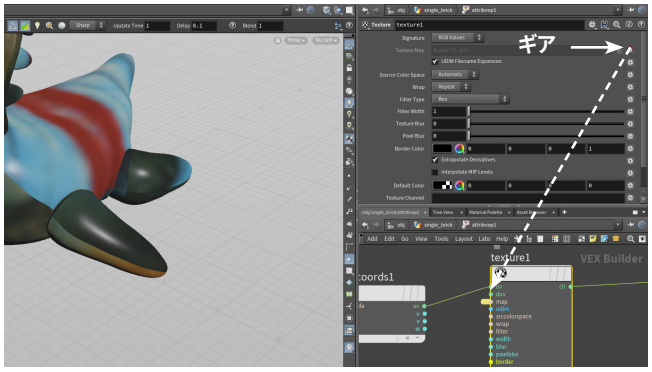
matchsize ノードの出力を、***attributevop*** ノードの 1 つ目の入力に接続します。この新しいノードに **Display フラグ**を設定します。

パラメータエディタで、**Run Over** を **vertices** に設定します。



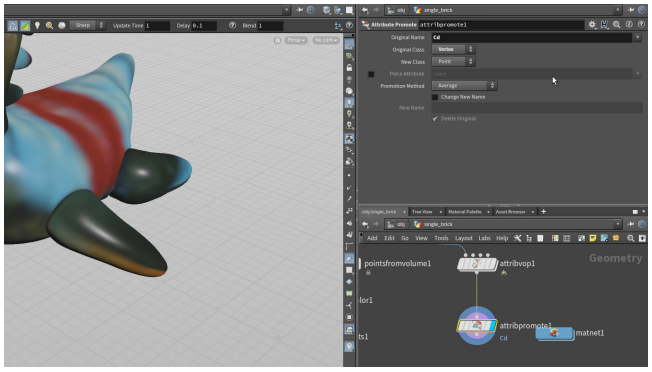
04 ***attributevop*** ノードを**ダブルクリック**して中に入り、**Tab** キーを使用して **Texture VOP** を追加します。それを ***geometryvopoutput*** ノードの **Cd** 入力に接続します。

UV coordinate ノードを追加して、***texture*** ノードの **UV** 入力に接続します。



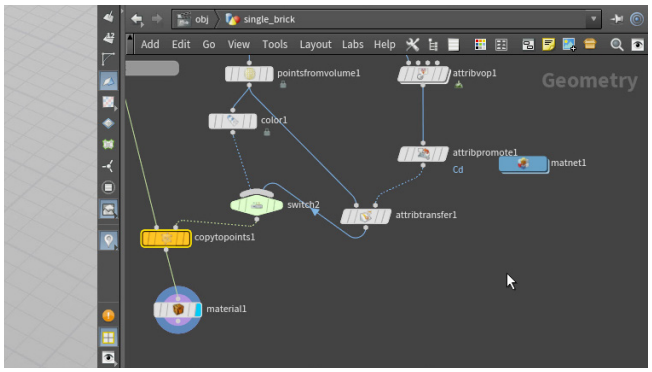
05 ***texture*** ノードを選択します。**Texture Map** パラメータの右端の**ギア** アイコンをクリックして、メニューから **Promote Parameter** を選択します。このパラメータがこのノードの上位レベルに追加されます。

マップの横にある小さいノブ(ツマミ)をクリックします。パラメータエディタで、**Label** を **Texture Map** に変更します。



06 **U を押して** 1 つ上のレベルに移動します。**Texture Map** パラメータが表示されています。ここでは、デフォルトの **Mandril. pic** テクスチャのままに設定しておきます。終端に ***toylowres*** テクスチャマップを追加します。

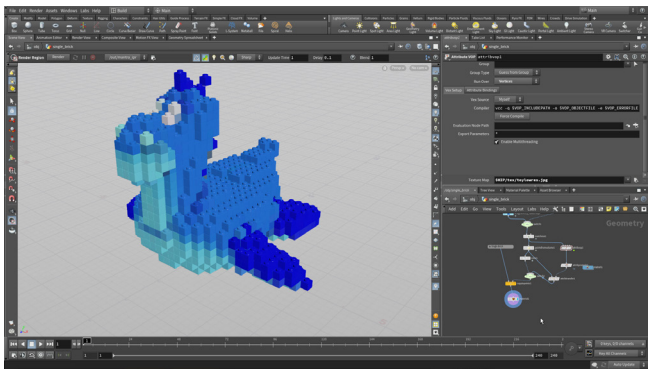
このチェーンの終端に **Attribute Promote** を追加します。**Original Name** を **Cd**、**Original Class** を **Vertex** に設定します。**New Class** の設定は **Point** のままにしておきます。



07 **Attribute Transfer** ノードを追加します。***pointsfromvolume*** を 1 つ目の入力に接続し、***attribpromote*** を 2 つ目の入力に接続します。**Attributes** タブで、**Points** フィールドの右側の矢印をクリックし、**Cd** を選択します。

color ノードの後に switch ノードを追加し、***attribtransfer*** ノードを switch に接続します。このノードの名前を ***texture_switch*** に変更します。

Switch を **1** に設定します。***copytopoints*** ノードに **Display フラグ**を設定します。これで、テクスチャマップからのカラーがコピーしたポイントに転送されるようになりましたが、ブロックが回転しています。**Transform Using Target Point Orientation** をオフにして、ブロックをまっすぐにします。



08 ***material*** ノードに **Display フラグ**を設定します。

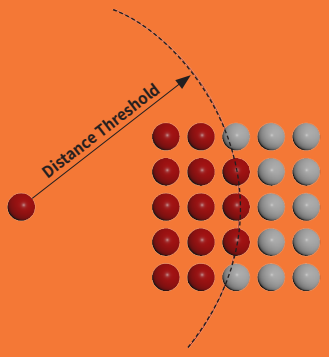
attribvop ノードを選択したら、**Texture Map** パラメータの右端の**ファイル選択**ボタンをクリックして、***toylowres.jpg*** テクスチャに移動し、**Accept** をクリックします。テクスチャマップのカラーが頂点に割り当てられているのを確認できます。

作業内容を**保存**します。

ATTRIBUTE TRANSFER

あるジオメトリから別のジオメトリにアトリビュートを転送したい場合、Attribute Transfer を使用します。このノードは、**Distance Threshold**(距離の閾値)などのパラメータを使用して、アトリビュートをコピーします。

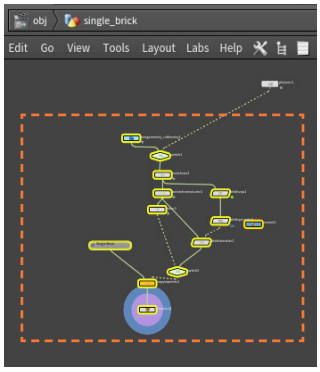
ゴムのおもちゃの場合は、ジオメトリ上のポイントの **Cd** アトリビュートを、ブロックをコピーするのに使用したポイントクラウドに転送しています。UV やキャプチャウェイトなどのアトリビュートもコピーできます。

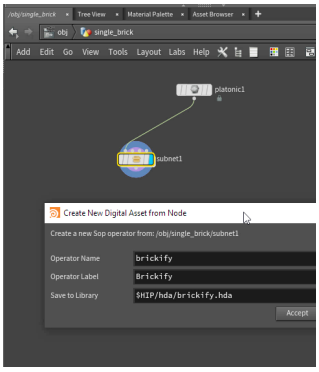


パート 5

ブロック化したデジタルアセットの作成

ブロック化のレシピが機能するようになり、ノードが適切に接続されました。次はいくつかのノードをまとめて、1 つの Houdini デジタルアセット（HDA）ノードを作成します。アセット内部からトップレベルにパラメータをプロモートしたネットワークを共有して、アセットが使用されるたびにユニークな結果を生成できるインターフェースを作成します。

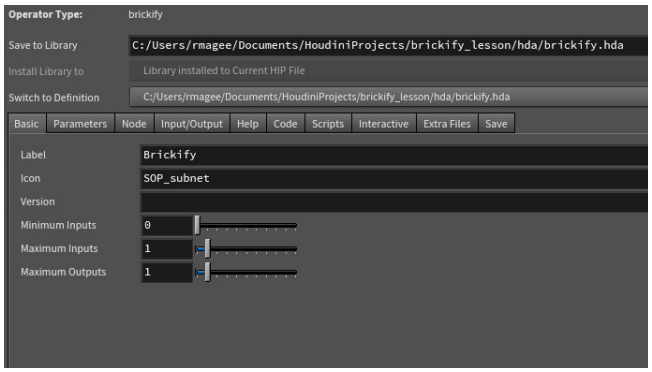




01

ネットワークビューで、***platonic*** ノードを横にドラッグします。ネットワークの他のすべてのノードを選択し、**Assets** メニューから **New Digital Asset From Selection...** を選択します。これで、単一のノードに折り畳まれます。

Operator Name を ***brickify*** に設定すると、**Operator Label** が ***Brickify*** に変更されます。**Save to Library** の右端のボタンをクリックします。Locations サイドバーで $HIP を選択し、***HDA*** ディレクトリをダブルクリックします。**Accept** を押し、**Create New Digital Asset** ダイアログで再度 **Accept** をクリックします。

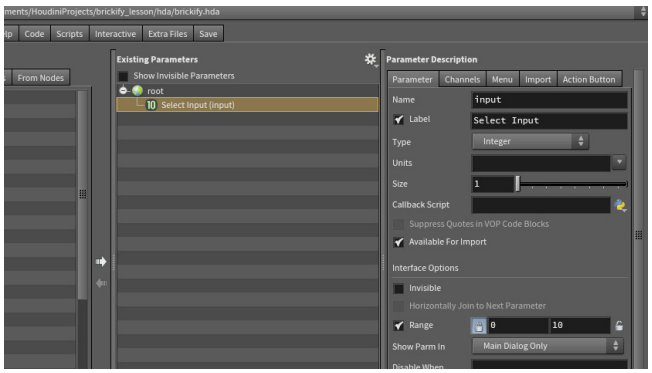


02

これにより **Type Properties** ウィンドウが表示されます。**Basic** タブが表示されていることを確認します。**Minimum Inputs** を **0** に設定して、入力がなくてもアセットが機能するようにします。**Maximum Inputs** パラメータを 1 に設定して、入力できるノードの数を定義します。

これは、platonic ノードに現在接続されている入力です。このデジタルアセットを後で使用する際は、この入力を使用して別の形状を指します。

Apply を押します。ウィンドウが閉じてしまうので、**Accept は押さないでください**。



03

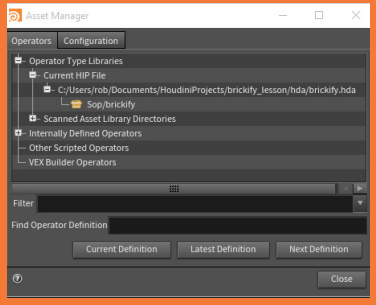
ネットワークビューで、新しいアセットノードを ***brickify_asset*** という**名前に変更**し、**ダブルクリック**して中に入ります。***testgeometry_rubbertoy*** と ***Subnetwork Input*** ノードを切り替えている ***switch*** ノードをクリックします。Subnetwork Input ノードは、上のレベルからプラトン立体のティーポット形状を取り込んでいます。

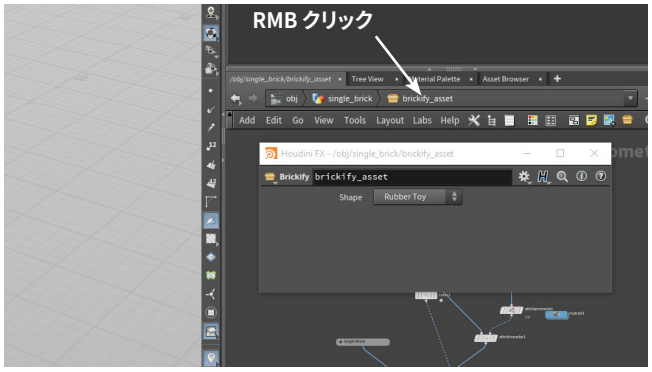
Type Properties ウィンドウで **Parameters** タブをクリックします。**Select Input** パラメータ名をクリックして、それを **Type Properties** ウィンドウの **Existing Parameters** リストに **LMB ドラッグ**します。それを root にドロップして、UI に追加します。**Apply** をクリックするとパラメータが追加されますが、Type Properties ウィンドウは終了しません。

.HDA ファイルとは？

アセットを保存すると、ディスク上に **.hda** ファイルが作成されます。HDA とは **Houdini Digital Asset** という意味で、アセット定義がこのファイル内に格納され、シーンに参照されます。このファイルには、ノード、プロモートされたパラメータ、UI 要素などの情報が含まれています。このファイルは、複数の人がさまざまなショットで参照し、共有できます。

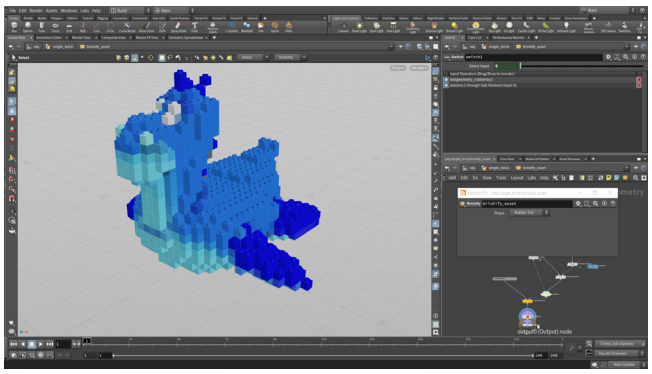
シーンに参照されているアセットを管理するには、**Assets** メニューで **Asset Manager...** を選択し、**Current HIP File** を開きます。





04 ネットワークビューのパスバーで ***brickify_asset*** を **RMB クリック**して、**Parameter and Channels > Parameters** を選択します。2 つの ***brickify*** アセットパラメータを含むフローティングパラメータウィンドウが表示されます。これらは、このアセットを使用する誰もが使用できるパラメータです。さらに追加していきましょう。

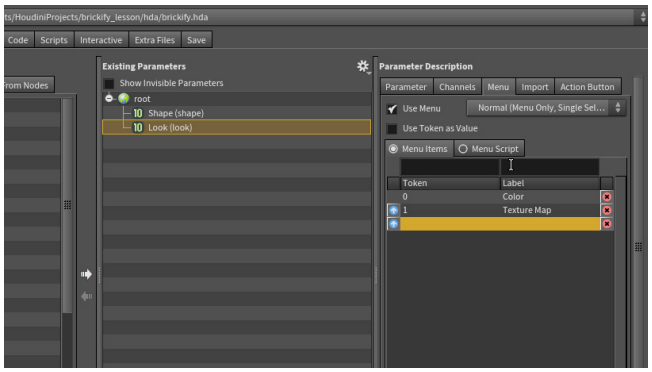
brickify ノードには新しいパラメータがあります。値を **0 から 1** に変更して、シーンへの影響を確認します。問題は、名前があまり適切でないことと、このケースではスライダよりもメニューのほうがずっと効率的に機能することです。そこで、Type Properties を使用して UI を微調整します。



05 パラメータリストで ***Select Input*** パラメータをクリックします。左側に、表示を微調整するためのオプションがあります。**Name** を ***shape*** に、**Label** を ***Shape*** に変更します。

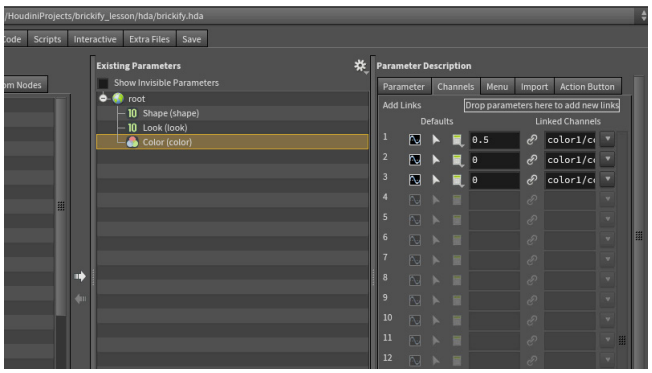
Menu タブをクリックして、**Use Menu** をオンにします。Menu Items で、**Token** に **0**、**Label** に ***Rubber Toy*** と入力して、Enter を押します。次に、**Token** に **1**、**Label** に ***Custom Shape*** と入力します。**Apply** を押します。

フローティングパラメータウィンドウには、**Shape** パラメータとメニューが表示されるようになります。実際に試して動作を確認しましょう。



06 ***color*** と ***attributetransfer*** ノードの下にある 2 つ目の ***switch*** ノードを選択し、**Select Input** パラメータをパラメータリストにプロモートします。**Name** を ***look*** に、**Label** を ***Look*** に変更します。

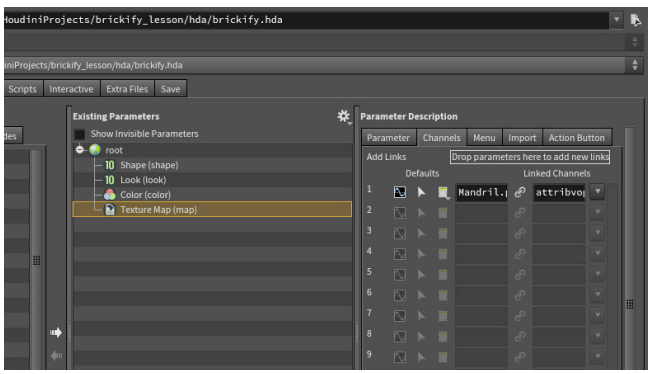
Menu タブをクリックして、**Use Menu** をオンにします。Menu Items で、**Token** に **0**、**Label** に ***Color*** と入力して、Enter を押します。次に、**Token** に **1**、**Label** に ***Texture Map*** と入力します。**Apply** を押します。



07 ネットワークで ***color*** ノードを選択し、**Color** パラメータをパラメータリストにプロモートします。これにより、カラーウィジェット、カラーホイール、RGB フィールドが表示されます。

Type Properties ウィンドウで **Apply** を押し、このパラメータが ***brickify_asset*** パラメータインターフェースでどのように見えるのかを確認します。

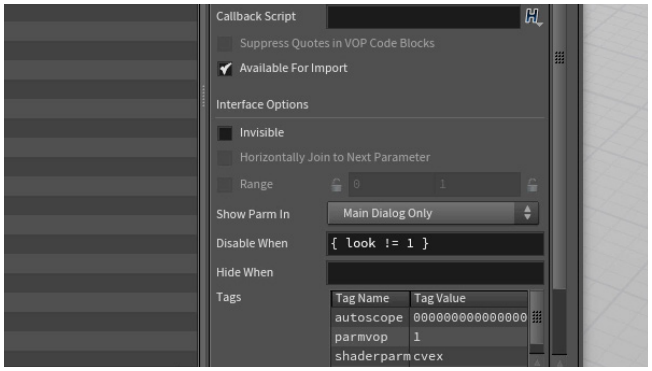
注: 間違って **Accept** を押した場合、新しいパラメータはアセットに保存されますが、**Asset** メニューを使用して **Edit Asset Properties... > brickify** を選択し、再度開く必要があります。



08 ***attributevop*** ノードを選択し、**Texture Map** パラメータをパラメータリストにプロモートします。

Parameter Description セクションで、**Channels** タブをクリックして**デフォルト**値を **Mandril.pic** に変更します。これは、パスを読み込む必要のないデフォルトのテクスチャマップで、より信頼性の高いデフォルトです。後で、***toylowres.jpg*** テクスチャマップを再読み込みします。

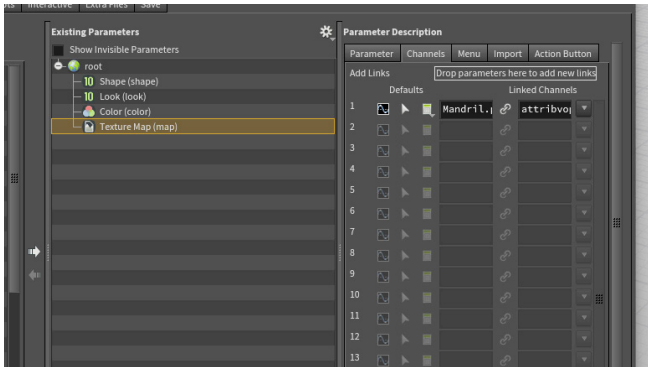
Type Properties ウィンドウで **Apply** を押すと、***brickify_asset*** パラメータインターフェースにこのパラメータが表示され、Mandril テクスチャマップによってブロックに色が付いています。



09 どのパラメータがどの形状に関連付けられているのかを確認するには、メニュー選択に基づいてパラメータを無効にしたり有効にします。**Color** パラメータをクリックし、**Disable When** フィールドに `{ look != 0 }` と入力します。

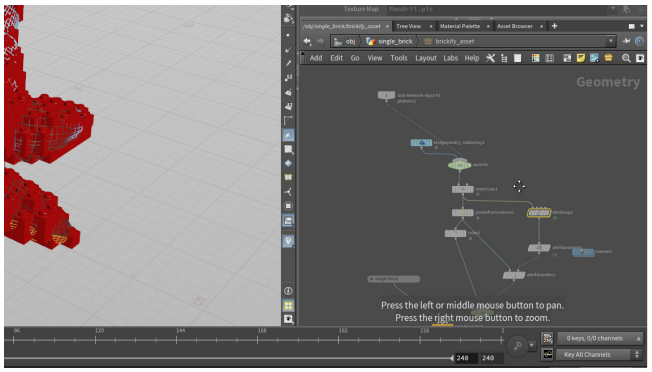
これにより、Look メニューで **Color** が選択されていなければ、このパラメータは無効になります。次に、**Texture Map** パラメータをクリックし、**Disable When** フィールドに `{ look != 1 }` と入力します。

Apply を押して、**Look** メニューを使用して結果をテストします。必要ない場合、パラメータが無効になっているのを確認できます。**Hide When** オプションを使用して非表示にすることもできますが、ここでは無効で問題ありません。



10 ***brickify_asset* テクスチャマップ**パラメータはデフォルトで **Mandril.pic** になっています。常に使用できるテクスチャマップで、汎用性の高いデフォルトです。おもちゃのマップに戻るには、**Texture Map** の横にあるファイル選択をクリックし、***$HIP*** を再度クリックしてから ***tex*** ディレクトリの中に入り、***toylowres.jpg*** ファイルを選択する必要があります。

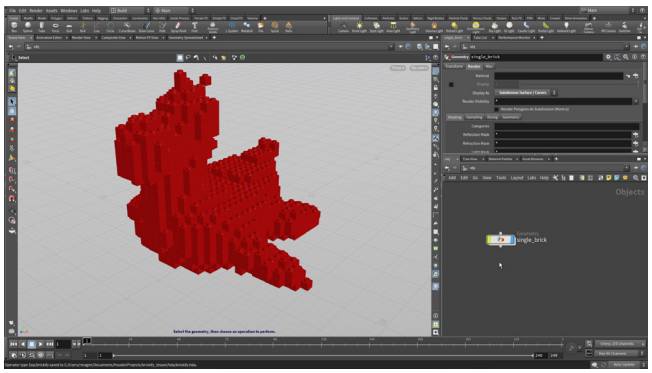
現在、アセットではデフォルトの ***Mandril.pic*** が使用されるようになっています。



11 **Accept** を押してアセットへの変更を保存し、Type Properties ウィンドウを閉じます。

ネットワークビューで、**U を押して** 1 つ上のレベルに戻ります。***brickify_asset*** ノードを選択した状態で、メインメニューから **Assets > Lock Asset > Brickify** を選択します。プロンプトが表示されたら、**Save Changes** を押します。

brickify_asset を**ダブルクリック**して中に入ると、ネットワークがグレーアウトされ、これらのノードが保護されていることが分かります。このアセットは、パラメータを介してのみ操作できます。内部の動作に変更を加えるには、アセットのロックを解除する必要があります。



12 オブジェクトレベルに移動します。シーンファイルを**保存**して、これまでの作業内容を保持します。

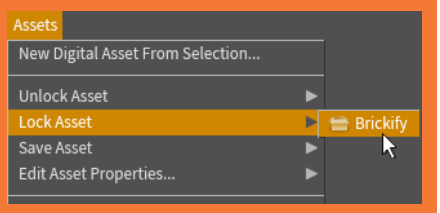
これで、シーンファイルと、シーンがアセットを作成するために参照する .hda ファイルができました。このライブラリを使用すると、このシーンでアセットの他のインスタンスを作成したり、アセットを別のシーンに追加することができます。

次のセクションでは、アセットをテストして、適切に動作するかどうか確認します。作業用のアセットを 1 つと、目的通りに動作するかどうかをテストするためのアセットを 1 つ用意しておくことをお勧めします。

アセットのロックとロック解除

Assets メニューを使用して、選択したアセットをロック/ロック解除できます。アセットが**ロック**されている場合、HDA ファイルが参照され、アセットの挙動が決まります。アセットが**ロック解除**されている場合、アクティブ定義はシーンファイル内にあります。アセットをロックしているとき、変更を加えた場合は保存するかどうかを確認されます。

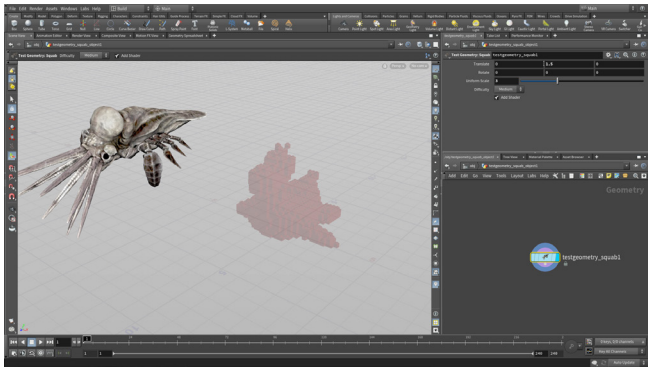
アセットのノードを **RMB クリック**すると、**Allow Editing of Contents** でアセットのロックを解除したり、**Match Current Definition** でアセットをロックできます。ただし、保存するかどうかは尋ねられず、変更が失われる可能性もあるため注意してください。



パート 6

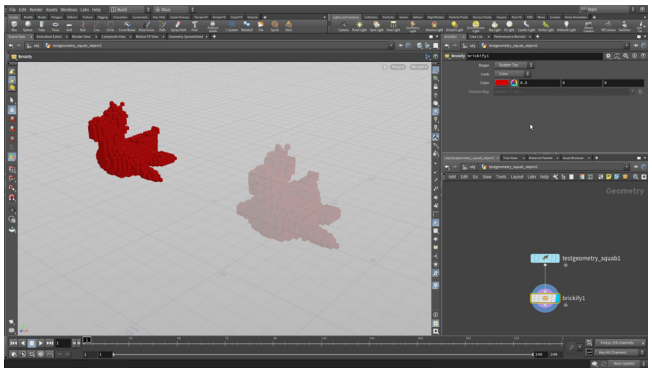
デジタルアセットのテスト

デジタルアセットは、単一のシーンファイルで複数回インスタンス化できます。異なるジオメトリでこのアセットを使用して、動作をテストしていきます。最初のアセットに設定された変更を素早く確認できるよう、テストバージョンを用意しておくことをお勧めします。アセットが適切に動作すれば、他のシーンファイルでも使用できます。



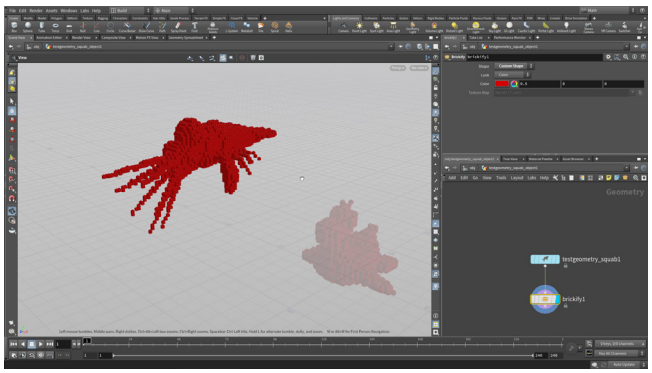
01 **Tab** キーを使用して、イカ蟹の**テストジオメトリ**を選択します。**Enter** を押して原点に配置したら、ハンドルを使用してゴムのおもちゃの横に移動します。

新しいオブジェクトノードを**ダブルクリック**して、ジオメトリレベルに入ります。ノードを**選択**して、**Scale** を **3**、**Translate Y** を **1.5** に設定します。これにより、イカ蟹がゴムのおもちゃよりも少し大きくなります。



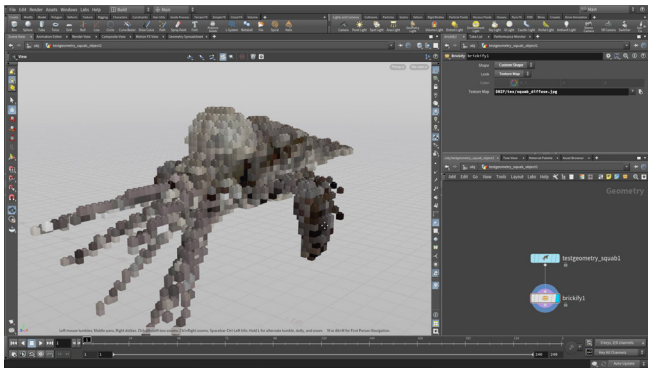
02 テストジオメトリノードの出力を **RMB クリック**して、**brickify...** と入力し、メニューから **brickify** アセットを選択します。これで、アセットがこの新しいネットワークに配置されます。

Display フラグを設定すると、**赤色**のゴムのおもちゃがもう 1 つ表示されます。こうなるのは、これらがこのアセットのデフォルトだからです。



03 ***brickify*** アセットノードで、**Shape** パラメータを **Custom Shape** に設定すると、ブロック化されたイカ蟹が表示されます。新しい形状がアセット内のノードネットワークを経て、ユニークな結果が生成されています。この形状が地面より少し上にあるのは、アセット内部に **Match Size** ノードがあるためです。

このようにすると、デジタルアセットをツールとして使用できます。複数のアクションを単一のノードにパッケージ化して、パイプラインに配置することができます。ワークフローが高速化するうえ、一貫した結果を実現しやすくなります。



04 ***testgeometry_squab*** ノードを選択します。**Asset** メニューから、**Edit Asset Properties > Squab** を選択します。**Type Properties** ウィンドウで、Extra Files タブをクリックし、***squab_diffuse.jpg*** を選択します。**Save as File** ボタンをクリックして、***tex*** フォルダに保存します。テクスチャがデジタルアセットに保存されたので、アセットと一緒に共有することができます。

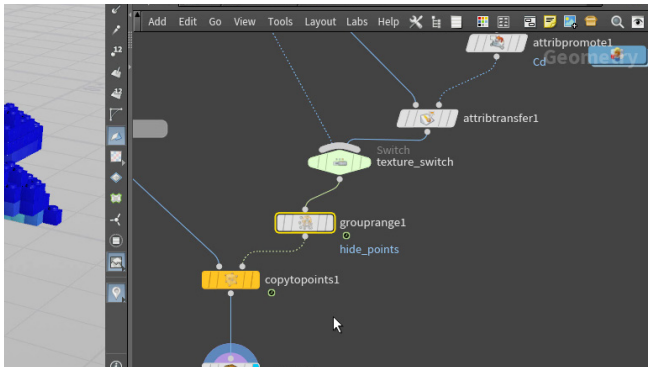
これで、ディスク上のこのテクスチャと、***brickify*** ノードの **Texture Map** パラメータを使用して、ブロックにカラーを追加できるようになりました。

完了したら、オブジェクトレベルに移動して、このオブジェクトに ***squab***、もう 1 つのオブジェクトに ***rubbertoy*** と**名前**を付けます。作業内容を**保存**します。

パート 7

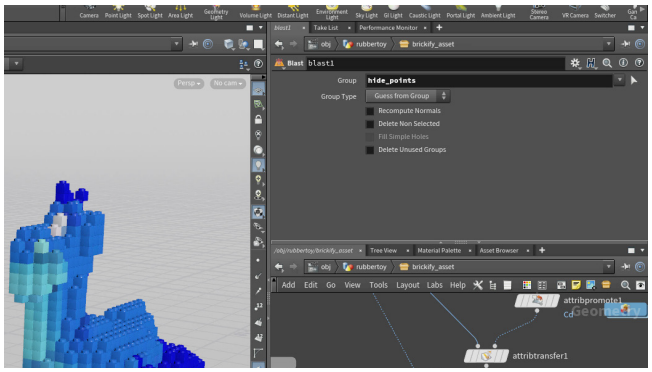
ブロックのアニメート

アセットには機能を追加していくことができます。ブロックが自動で積み上がるアニメーションを作成しましょう。
このためには、ネットワークにさらにノードを追加して、アセットにこの新しい機能を含める必要があります。
結果を .hda ファイルに保存したら、誰でもこのアセットを使ってその機能を使用できるようになります。



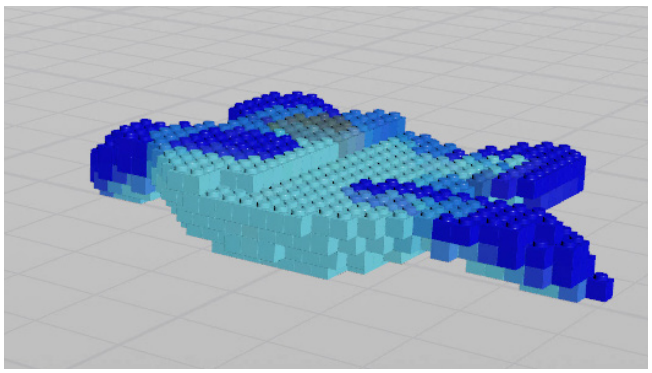
01 ***squab*** オブジェクトを非表示にして、***rubbertoy*** オブジェクトの中に入ります。***brickify_asset*** を選択し、**Assets > Unlock Asset > Brickify** を選択します。***brickify_asset*** ノードを**ダブルクリック**したら、***texture_switch*** の出力を **RMB クリック**して、**Group by Range** を選択します。そのノードを配置して **Display フラグ**を設定したら、次のようにパラメータを設定します。

- **Group Name** を ***hide_points*** にする
- **Group Type** を **Points** にする
- **Range Type** を **Start and Length** にする
- **Length** を ***($F-1)*20*** にする
- **Range Filter** で、**Select** を **1**、**Of** を **1** のままにしておく



02 ***grouprange*** ノードの出力を **RMB クリック**し、**Polygon > Blast** を選択します。このノードを配置したら、**Group** の横の矢印を使用して、***hide_points*** グループを選択します。**Delete Non Selected** をオンにして、グループの外側のポイントを削除します。***blast*** ノードに **Display フラグ**を設定します。

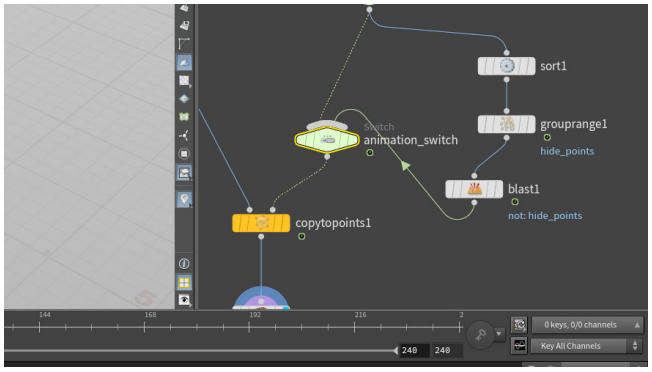
Play を押して、フレーム毎に増えるポイントを確認します。チェーンの終端の ***material*** ノードに **Display フラグ**を設定して、時間とともに増えるブロックを確認します。



03 現在は、ブロックが地面からでなく片側から現れています。これは、ポイントがポイント番号を基準に表示されるようになっているからです。これを制御するには、ポイントの順番を変更して目的のルックにする必要があります。

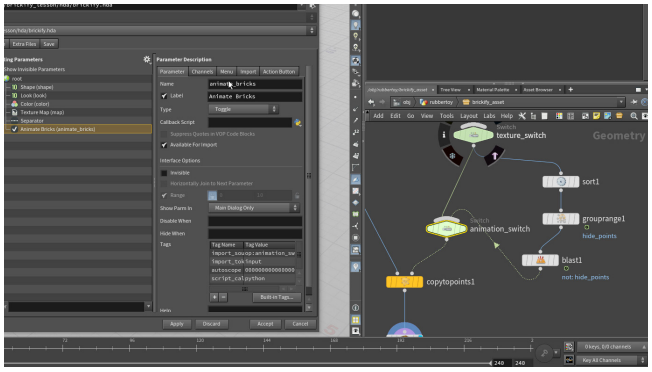
texture_switch ノードの出力を **RMB クリック**し、**Sort...** と入力していき、**Sort ツール**を選択します。このノードを配置したら、**Point Sort** を **Along Vector** に変更します。これを **0, 1, 0** に設定すると、ポイントが下から上に向かって現れるようになります。

再生して結果を確認します。さまざまなベクトルをテストして、アニメーションへの影響を確認してください。



04 ブロック化エフェクトをアニメートしたいかどうかを選択できるように、もう 1 つ switch ノードを追加します。ネットワークビューで、**Tab を押して Switch...** と入力していきます。**Switch** を選択して、ノードを配置します。ノードの名前を ***animation_switch*** に変更します。

texture_switch ノードの出力をクリックして、***switch*** ノードの入力に接続します。***blast*** ノードでも繰り返します。これにより、元の形状が最初のオプション、アニメーションエフェクトが 2 番目のオプションになります。**Select Input** を **1** にすると、アニメートされたブロックが表示されますが、ここでは **0** にしておきます。



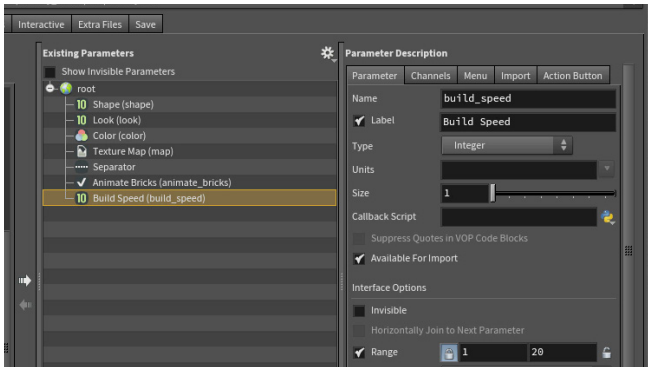
05

Assets メニューから、**Edit Asset Properties > Brickify** を選択します。**Parameters** タブに移動して、**Create Parameters** セクションの **Separator** をリストの下部にドラッグします。

次に、***animation_switch*** ノードの **Select Input** をパラメータエディタから新しい Separator の下にドラッグします。**Name** を ***animate_bricks*** に、**Label** を ***Animate Bricks*** に設定します。次に、**Type** を **Toggle** に変更します。これで、**オン (1) またはオフ (0)** の設定に制限されます。

Parameter Description セクションで、**Channels** タブをクリックしてデフォルト値を **0 (オフ)** に設定します。

Accept をクリックして変更を保存します。



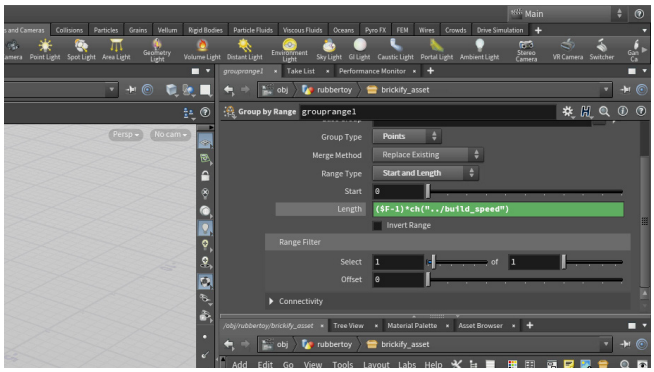
06

Type Properties の **Create Parameters** セクションから、**Integer** パラメータを ***animate_bricks*** パラメータの下にドラッグします。**Name** を ***build_speed*** に、**Label** を ***Build Speed*** に設定します。

Range オプションをオンにして、最初の値を **1**、2 番目の値を **20** にします。1 の横の**錠**アイコンをクリックすると、値が 1 よりも小さくなりません。

Parameter Description セクションで、**Channels** タブをクリックしてデフォルト値を **1** に設定します。

Accept を押して保存し、ウィンドウを閉じます。

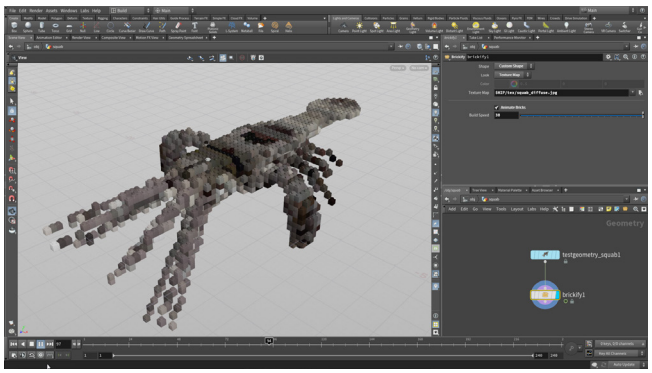


07

このパラメータはまだどこにも取り付けられていませんが、これを使用して、***grouprange*** ノードの **Length** エクスプレッションを駆動させることができます。新しい ***grouprange*** ノードを選択し、**Length** エクスプレッションを `($F-1)*ch("../build_speed")` に変更します。

ネットワークの終端に **output** ノードがあります。これにより、ネットワークの別のノードの **Display フラグ**がオンになっていても、アセットの適切な出力を取得できます。

brickify_asset ノードを選択したまま、メインメニューから **Assets > Save Asset > Brickify** を選択します。Type Properties ウィンドウを再度開くことなく、.hda ファイルにこのエクスプレッションを保存できます。



08

brickify_asset ノードを選択したまま、メインメニューから **Assets > Lock Asset > Brickify** を選択します。これで、ブロック化エフェクトをカスタムツールにまとめることができました。このツールは、アーティストたちがさまざまなショットで使用することができます。

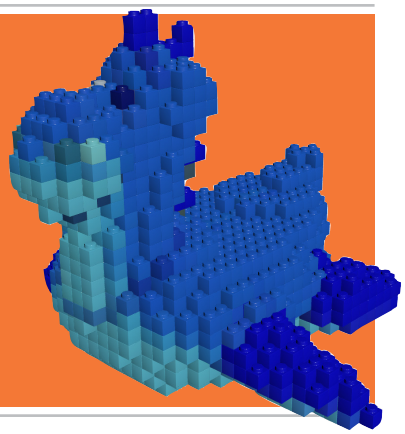
Squab ネットワークに移動すると、***brickify*** で新しい機能が使用できるようになっています。**Animate Bricks** トグルをオンにしたら、この形状はブロックの数が多いので、**Build Speed** を約 **30** に設定します。

再生して結果を確認します。

まとめ

アーティストが利用しやすいノードベースのワークフローを使用して、スクリプトを記述することなく、共有可能なツールを作成できました。HDA をディスクに保存することで、ディスク上のアセットとなり、複数のショットで参照できるようになります。

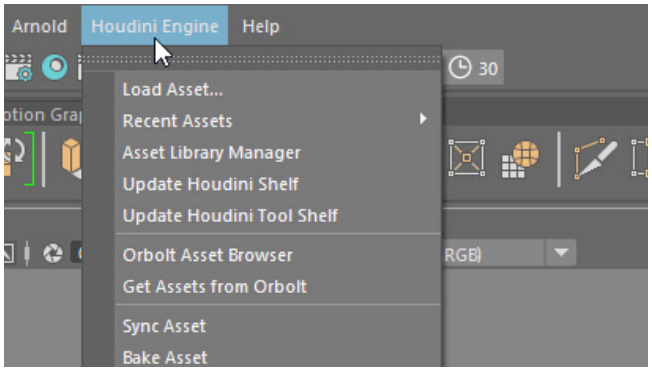
Houdini デジタルアセットは、アーティストがこうしたツールを効果的に共有できるようにするものであり、スタジオレベルのプロダクションを支えます。プロシージャルアセットは反復作業の自動化が簡単にできるため、プロジェクトのクリエイティブなニーズに集中できます。



パート 8

他のアプリケーションに HDA を読み込む

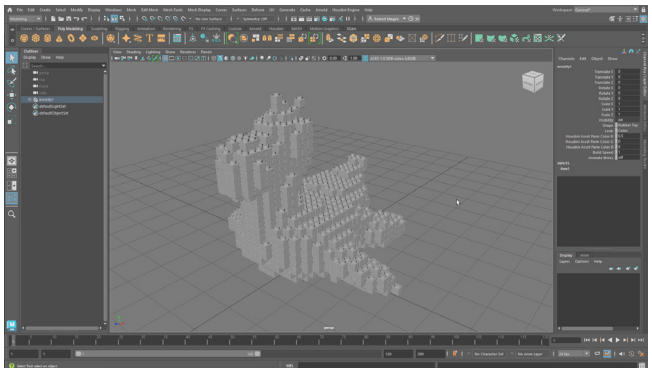
Houdini デジタルアセット（HDA）をディスク上に保存したら、Houdini Engine プラグインを使用して、そのアセットをホストアプリケーションに読み込むことが可能です。こうしたプラグインを使ってアセットを共有すれば、同僚は Autodesk Maya や 3ds Max などの 3D アプリケーションや、Unity や Unreal Engine などのゲームエディタに直接アセットを読み込めるようになります。



01 ホストアプリケーションで Houdini Engine を使用するには、Houdini インストーラまたは Launcher を使用して、プラグインをインストールすることからはじめます。これによりプラグインが使用可能になりますが、さらに、セッション内で Houdini Engine を使用できるようにする手順が必要な場合があります。

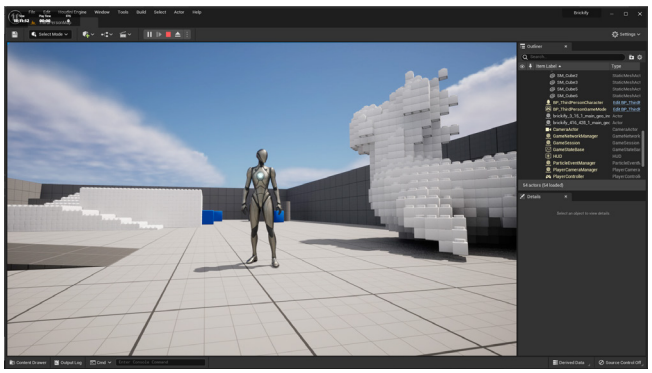
詳細は、**SideFX.com/engine** を参照してください。

Engine プラグインのタブをクリックしてから目的のプラグインをクリックして、詳細を確認します。インストールすると、ホストアプリケーションに Houdini Engine メニューが表示されるずです。メニューからアセットを読み込むことができます。



02 プラグインのインストールが完了したら、**Houdini Engine** メニューを使用してアセットを読み込むことができます。***brickify*** アセットがビューポートに読み込まれ、アセットパラメータを使用して操作できるようになります。

また、Maya または 3ds Max 向けにアニメーションをオンにすると、タイムラインを使用してシーケンスを再生できます。



03 Unreal では、**Import** ボタンを使用して、デジタルアセットをシーンに取り込みます。また、**Shape** を **Custom Shape** に設定して、ホストアプリケーション内でアセットをジオメトリに接続すれば、そのオブジェクトにブロック化が適用されます。

ブロックのカラーに何が起きたのか

各種ホストアプリケーション内で、ブロック化された形状に色が付けられていないことに気付くはずです。これは、プラグインと Houdini で情報の処理方法が必ずしも同じではないことが原因です。ポイントカラーはアセットに含まれていますが、ポストアプリケーションはその情報を受け取りません。

また、Maya および 3ds Max ではアニメーションが機能しますが、Unity や Unreal Engine では機能しません。Houdini アセットはゲームの実行体験には関与せず、組み込まれたアニメーションは無視されるからです。ホストアプリケーションの機能に合わせて、アセットを調整することが重要です。

HOUDINI FOUNDATIONS

ワイングラスの粉砕

このレッスンでは、ワイングラスを粉砕した後、時間を操作して液体（ワイン）を空中に保持します。このエフェクトでは、グラスの粉砕に RBD シミュレーション、ワインに流体シミュレーションを使用します。ダイナミクスネットワークをセットアップする方法と、シミュレーションを出力する方法を学習できます。ビジュアルエフェクトショットではたいてい、さまざまな種類のダイナミクスソルバを組み合わせて使用します。Houdini のダイナミクスネットワークは、多様なソルバを使い、それらによって 1 つのまとまった結果を得られるように設計されています。

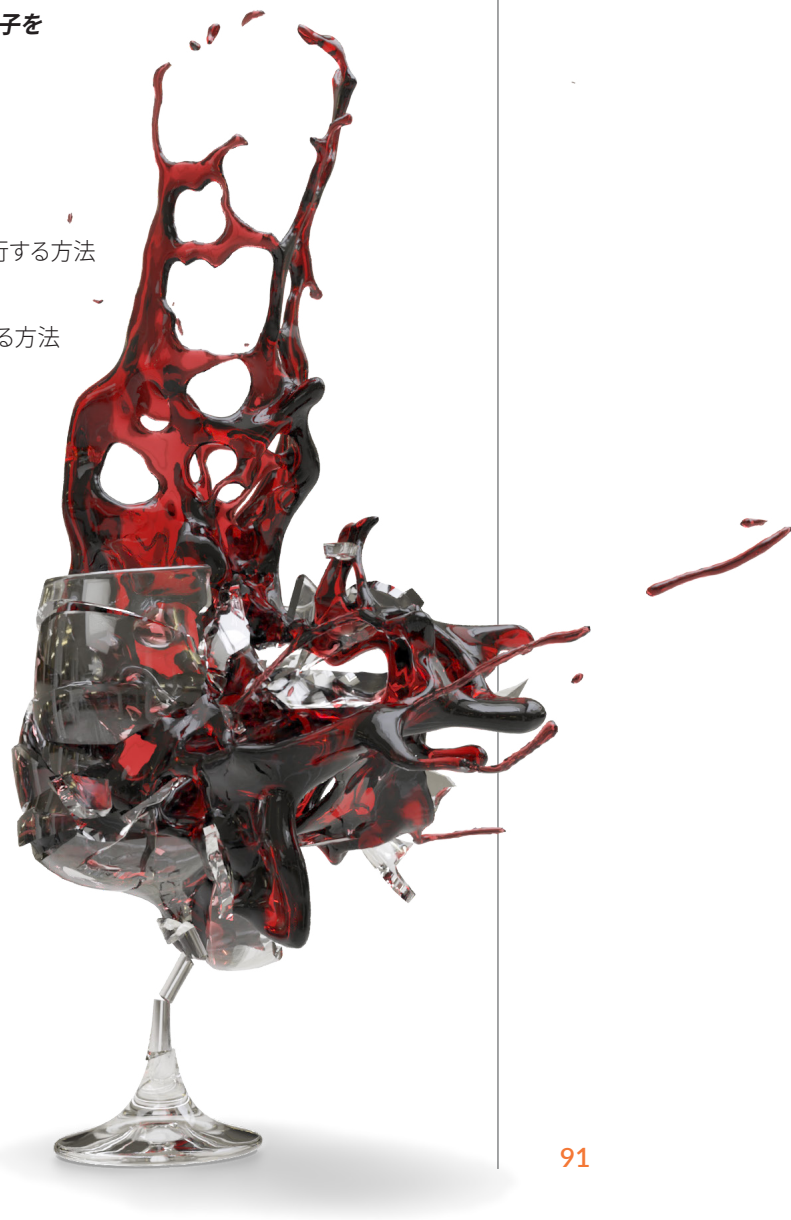
また、Retime ノードを使用して、爆発が最大のときにシミュレーションの速度を落としてから、時間を逆戻りさせて始点に戻します。次にシミュレーションを Solaris/LOPS に移動し、ライトとカメラをセットアップした後、Karma レンダラを使用してショットをレンダリングします。

レッスンの目標

弾丸がワイングラスに当たってグラスが割れ、液体が飛び散る様子をシミュレートします。

学習内容

- **ワイングラス、弾丸、液体**のジオメトリをモデリングする方法
- **ブーリアン**を使用してガラスジオメトリを**事前に破壊**する方法
- グラスを粉砕する弾丸の**リジッドボディシミュレーション**を実行する方法
- 液体が飛び散る **FLIP 流体シミュレーション**を実行する方法
- シミュレーションを**リタイム**し、速度を落としてから、逆行させる方法
- **Solaris/LOPS** で使用するために、結果を **USD** としてエクスポートする方法
- **Solaris/LOPS** で**ライト**と**マテリアル**をセットアップする方法
- **Karma** を使用して最終ショットを**レンダリング**する方法



使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、以下の Houdini 製品で実行可能です。

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

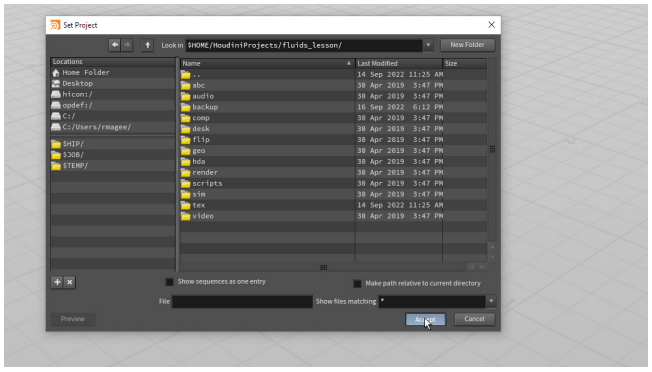
ドキュメントバージョン 4.0.1J | 2023 年 8 月

パート 1
ワイングラスのモデリング

まず、ポリゴンカーブを描画して回転させ、ワイングラスを作ります。Creasing（折り目）を使用してエッジを鋭くしたら、細分化して粉砕用にジオメトリの密度をあげます。その後、ワイングラスからジオメトリを抽出し、液体のシミュレーションに使用する形状を作成します。

プロジェクトファイル

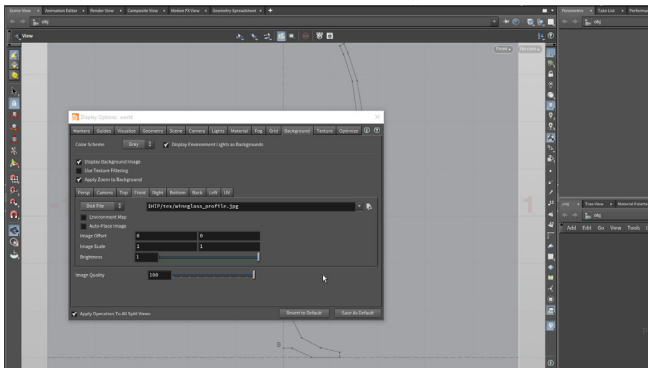
SideFX.com の流体チュートリアルのページ（このドキュメントを入手した場所です）から、***fluids_lesson_start*** ディレクトリをダウンロードします。名前を ***fluids_lesson*** に変更し、**Houdini Projects** ディレクトリに配置してください。



01

File > Set Project を選択します。***fluids_lesson*** ディレクトリに移動し（上記の説明を参照）、**Accept** を押します。これにより、このショットに関連するすべてのファイルがこのプロジェクトディレクトリとそのサブフォルダに集められます。

File > Save As... を選択します。新しい ***fluids_lesson*** ディレクトリが表示されているはずです。ファイル名を ***wineglass_01.hip*** に設定し、**Accept** をクリックして保存します。これで、***Tex*** フォルダのリファレンス画像にアクセスできるようになります。



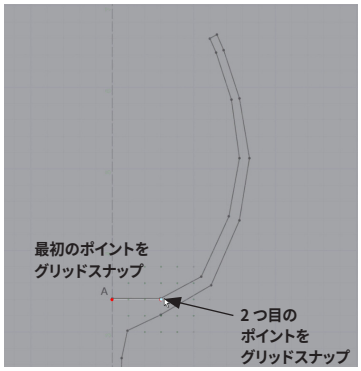
02

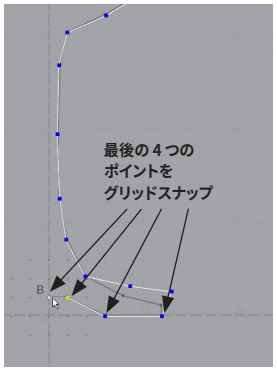
Scene View で、**V を押して** Radial メニューを表示し、**Viewport Layout > Four Views** を選択します。カーソルを ***Front*** パネルの上に移動して、**スペースバー + B** を押して拡大します。

マウスがビューポート上にある状態で **D** を押します。**Background** タブをクリックし、**Front** タブで、ファイルピッカーを使用して ***$HIP***、それから ***tex>wineglass_profile.jpg*** に移動します。次のように設定します。

- **Auto-Place Image** が**オフ**であることを確認する
- **Image Offset** を **0, 3** にする
- **Image Scale** を **5, 5** にする

これで、ビューをパンおよびドリーすると、背景も一緒に動きます。



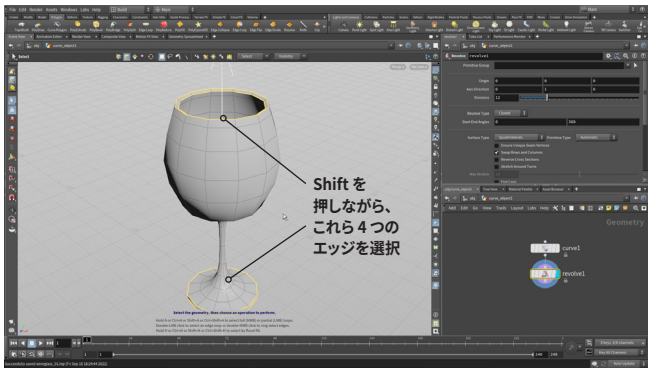


03

Polygon シェルフで、**Curve Polygon** ツールをクリックします。これにより、**Primitive Type** が **Polygon** に設定された **Curve** ノードが作成されます。**X を押して Grid** を選択し、グリッドスナッピングを有効にして、**ポイント A** をクリックします。次に、同じようにグリッドポイントの上にある 2 つ目のポイントをクリックします。**グリッドスナッピング**を**オフ**にして、画像のトレースを続けます。

グラスのプレート部分（土台）の最後の 4 つのポイントでは、ずれないように**グリッドスナッピング**を**オン**に戻します。ポイント B で終了したら、**Enter** を押してカーブを完了します。**グリッドスナッピング**を**オフ**にします。

オペレーションコントロールツールバーの **Edit モード**ボタンをクリックすると、ずれたポイントを移動できます。

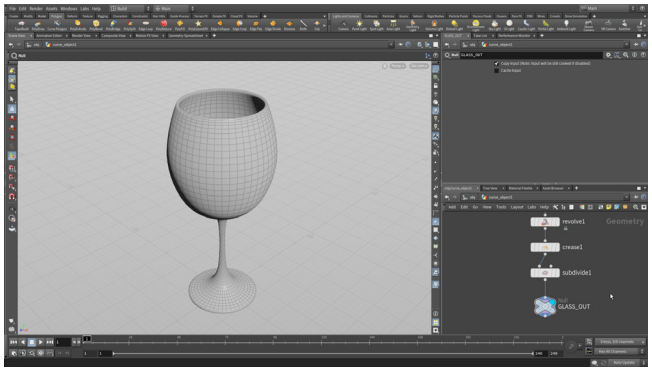


04

スペースバー + B を押して、4 ビューのレイアウトに戻ります。次に、マウスをパースビューの上に移動し、再度**スペースバー + B** を押して拡大します。これで、カーブが 3D で表示されるようになりました。

S を押して Select ツールにしたら、**N を押して**カーブ上のすべてのプリミティブを選択します。**C を押して**、**Model > Curves > Revolve** を選択します。これで、ワイングラスのモデルになります。

3 を押してエッジの選択に移り、カップの最上部（リム）のエッジを**ダブルクリック**します。**Shift** を押して、上から 2 番目のエッジとプレート（土台）の 2 本のエッジを**ダブルクリック**します。**Tab > Crease** を押して、**Crease** を **0.75** に設定します。

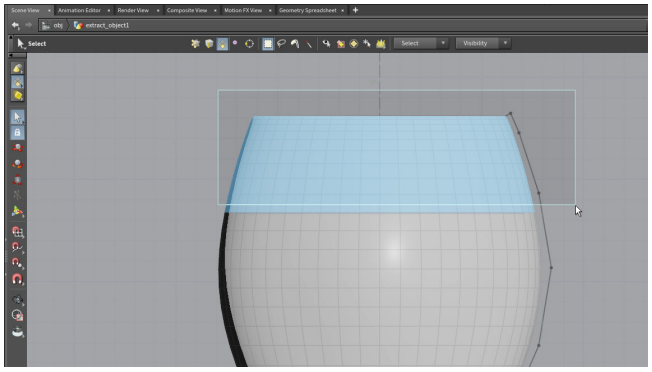


05 **4 を押して**、プリミティブ選択に切り替え、**N を押して**すべて選択します。次に **Tab > Subdivide を押します**。**Depth** を **2** に設定します。

これにより、モデルが細分化されます（折り目が付けられたエッジは他の領域よりもシャープになるようセットアップされています）。**Crease Weight** を高くするとエッジは尖ります。このグラスでは、ソフトな見た目の方がうまく機能します。

ネットワークエディタで、チェーンの終端に **Null** ノードを追加して、その Display フラグを設定します。このノードの名前を ***GLASS_OUT*** に変更します。

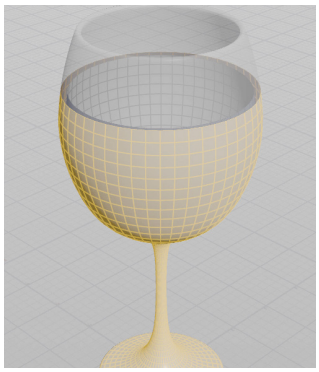
オブジェクトレベルに戻り、オブジェクトの名前を ***wine_glass*** に変更します。

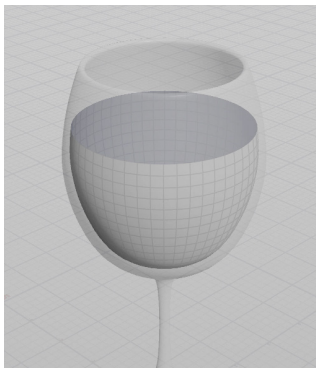


06 ***wine_glass*** ノードを選択した状態で、**N を押して**すべてのプリミティブを選択します。次に、**Modify** シェルフに移動して、**Extract** ツールをクリックします。これにより、ワイングラスジオメトリの ***objectmerge*** が作成されるので、それを新しいオブジェクトに配置します。

Front ビューから、ワイングラスの上部の面を**選択**して、**Delete** を押します。すると、ネットワークに面を削除する ***Blast*** ノードが追加されます。

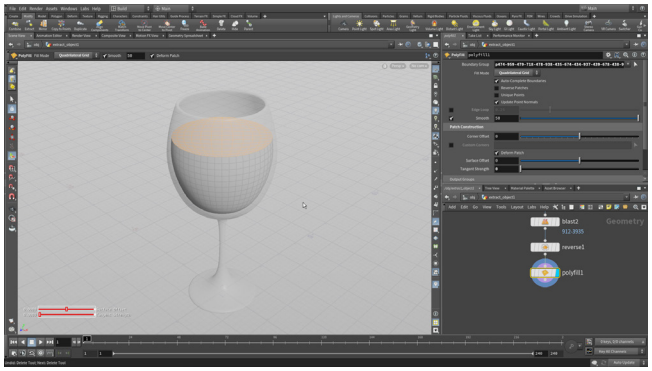
注：オリジナルのワイングラスのゴースト（半透明）表示バージョンが見えるのは、Scene View が **Ghost Other Objects** に設定されているからです。この設定は、作業状況の前後関係を確認できるので便利です。





07 ここでパースビューに戻って、カップの下部を**ダブルクリック**し、**Delete** を押します。これにより、2 つ目の Blast ノードが追加されます。これで、液体にしたい内側の面が残ります。ワインジオメトリの面は、外側が暗く見えます。これは、そちら側がプリミティブの後面側ということです。

N を押してすべてのプリミティブを選択します。**Tab > Reverse を押して**法線を外向きに反転させます。これで、各プリミティブの暗い側が形状の内側になります。

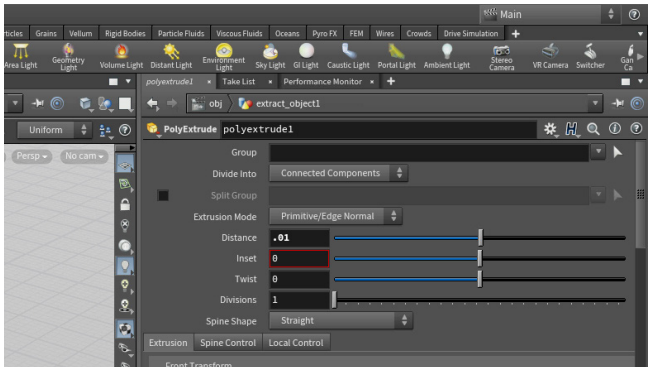


08 **3 を押して**エッジの選択に変更したら、形状のエッジをダブルクリックして、開いたエッジを選択します。**Tab を押して polyfill** と入力します。**Polyfill** がハイライトされたら、**Enter** を押します。

次のように設定します。

- **Fill Mode** を **Quadrilateral Grid** にする
- **Tangent Strength** を **0** にする

これで閉じた形状が作成され、レッスンの後半で使用する FLIP 流体のソースになります。



09 **4 を押して**、プリミティブ選択に切り替え、**N を押して**すべてのプリミティブを選択します。**C を押して**、**Model > Polygons > PolyExtrude** を選択します。**Distance** を **0.01** に設定してワイングラスとのオーバーラップを作成すると、液体を適切にレンダリングできます。

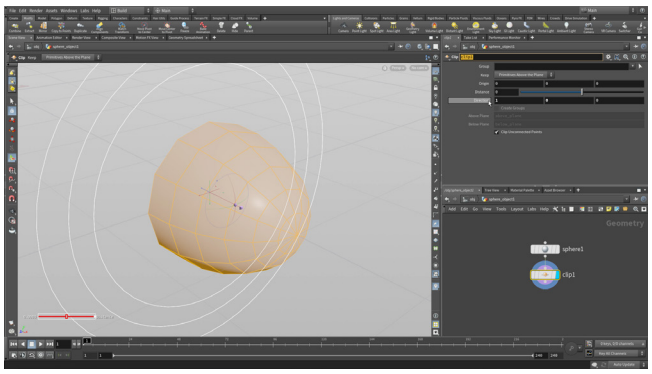
ネットワークエディタで、チェーンの終端に **Null** ノードを追加して、その **Display フラグ**を設定します。このノードの名前を ***FLUID_OUT*** に変更します。

オブジェクトレベルに上がり、このオブジェクトの名前を ***wine*** に変更します。

パート 2

弾丸のモデリング

弾丸のジオメトリを作成するためには、プリミティブの球から始め、それを半分にスライスします。次に、開口部を PolyExtrude で押し出したら、PolyFill で四角形トポロジを使用して形状を閉じます。その後、細分化して最終的な形状を定義します。このオブジェクトは非常に素早く移動するため、多くのディテールは必要ありません。

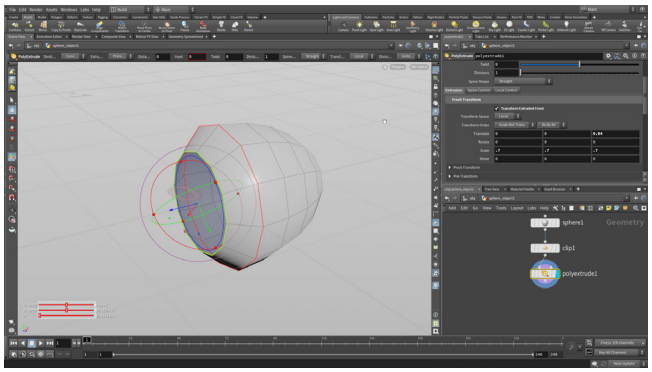


01

オブジェクトレベルに移動し、ネットワークビューで、wine および ***wine_glass*** オブジェクトの **Display フラグ**をオフにします。Scene View で、**C を押して**、**Create > Geometry > Sphere** を選択します。**Enter** を押して原点に配置したら、中に入って次のように設定します。

- **Orientation** を **X Axis** にする
- **Radius** を **0.2, 0.125, 0.125** にする
- **Columns** を **12** にする

Scene View で **Tab > Clip** を押したら、**N を押して**すべてのプリミティブを選択し、**Enter** を押して、**Direction** を **1, 0, 0** に設定します。



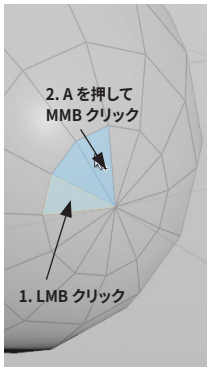
02

Select ツールをクリックしたら、**3 を押して**エッジ選択を有効にし、球の開口部を**ダブルクリック**します。**C を押して**、**Model > Polygons > PolyExtrude** を選択します。

Extrusion タブで、次のように設定します。

- **Transform Extruded Front** を**オン**にする
- **Translate Z** を **0.04** にする
- **Scale** を **0.7, 0.7, 0.7** にする

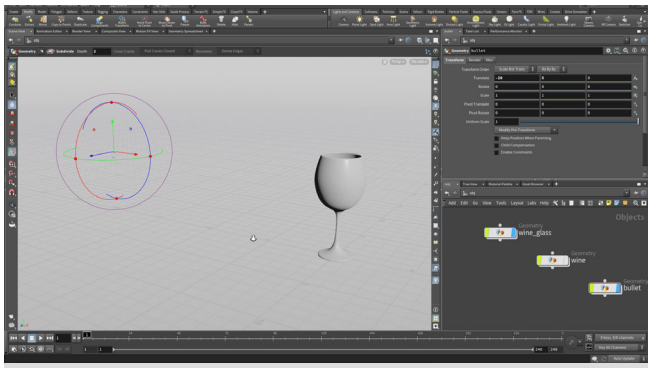
これにより少しジオメトリが追加されます。この後、弾丸の前面の三角形を削除して、形状を閉じます。



03

タンブルして、**S を押して** Select ツールにし、**4** を押して **面 / プリミティブ**選択にします。弾丸の先端で三角形の 1 つを選択したら、**A キーを押しながら** 2 つの三角形を **MMB クリック**して、三角形の面をすべて選択します。**Delete** キーを押して、それらを削除します。これにより、ネットワークに ***Blast*** ノードが追加されます。

ネットワークビューで、**Tab > PolyFill** を押して、***blast*** の後にそのノードを追加します。**Fill Mode** を **Quadrilateral Grid**、**Corner Offset** を **1** に設定し、**Display フラグ**をオンにします。これにより、弾丸の両開口部が適切な四角形トポロジで閉じられます。



04

最後に **Subdivide** ノードを追加して、**Depth** を **2** にして、**Display フラグ**を設定します。

ネットワークエディタで、チェーンの終端に **Null** ノードを追加して、その **Display フラグ**を設定します。このノードの名前を ***BULLET_OUT*** に変更します。

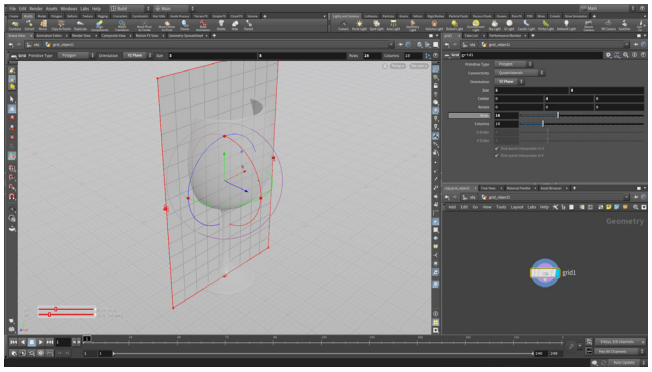
オブジェクトレベルに戻り、このオブジェクトの名前を ***bullet*** に変更します。***wine_glass*** の **Display フラグ**をオンにします。弾丸を **X** 軸で **-20**、**Y** 軸で **5**、**移動**させます。***Front*** 正射投影ビューに戻り、ワイングラスとの衝突ポイントが意図した通りかどうかを確認するとよいでしょう。

作業内容を**保存**します。

パート 3

ワイングラスの破砕

ワイングラスのひび割れを定義するために、Draw Curve ツールを使用して自然に見える線を作成し、それらを押し出してジオメトリのシートにします。次に、Mountain ツールを使用してサーフェスにノイズを加えます。その後、この混沌とした見た目の形状をワイングラスオブジェクトに結合し、シートを使用してグラスを粉砕するブーリアン演算をセットアップします。

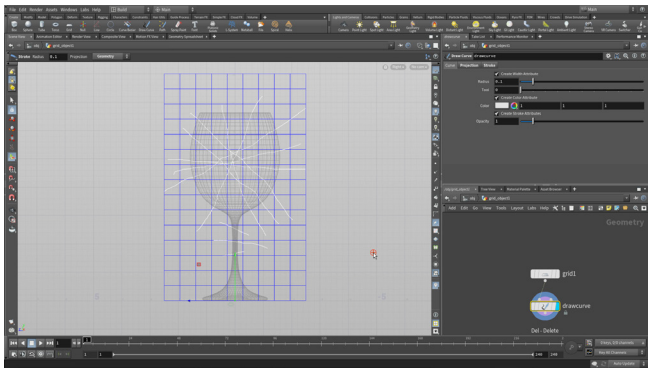


01

Scene View でオブジェクトレベルに移動し、**C を押して**、**Create > Geometry > Grid** を選択します。**Enter** を押して原点に配置したら、**I** で中に入って次のように設定します。

- **Orientation** を **YZ Plane** にする
- **Center Y** を **4** にする
- **Size X** を **5** にする
- **Size Y** を **8** にする
- **Rows** を **16** にする

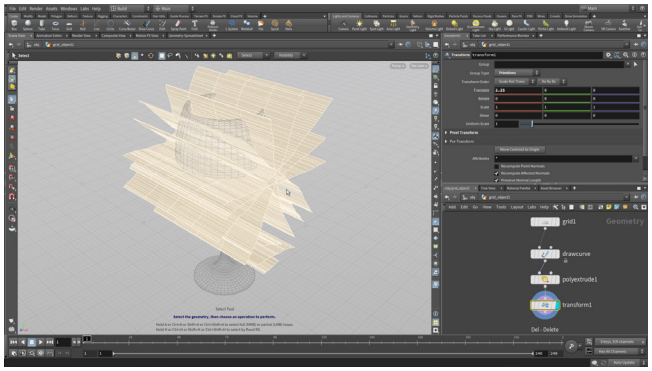
これにより、ワイングラスの形状に一致する、**Draw Curve** ツールのための描画サーフェスが作成されます。



02

Right ビューに移動し、**V を押して Shading > Wireframe** を選択し、ワイヤーフレームモードにします。グリッドとワイングラスの背後に弾丸が見えます。**N を押して**グリッド全体を選択し、**Create** シェルフの **Draw Curve** ツールをクリックします。

ワイングラスの上に、弾丸がガラスに当たる場所で交わるカーブを何本も描画します。グラスのステム（脚）も粉々になる予定なので、ステムを横切るカーブも追加します。不要なカーブを描いてしまったら、いつでも **Ctrl + Z** を押して取り消せます。



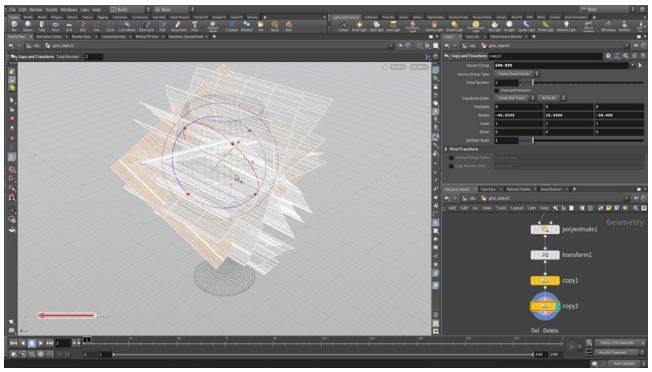
03

Select ツールに移動し、**N を押して**すべてのカーブを**選択**します。次に、**C を押して**、**Model > Polygons > PolyExtrude** を選択します。**Divisions** を **4** に設定します。

Extrusion タブで、次のように設定します。

- **Transform Extruded Front** を**オン**にする
- **Transform Space** を **Global** にする
- **Translate X** を **-4.5** にする

次に、別の箇所をクリックしてからすべてのジオメトリを選択し、**Tab > Transform** を押します。**Translate X** を **2.25** に設定して、このジオメトリを原点周辺の中央に配置します。



04

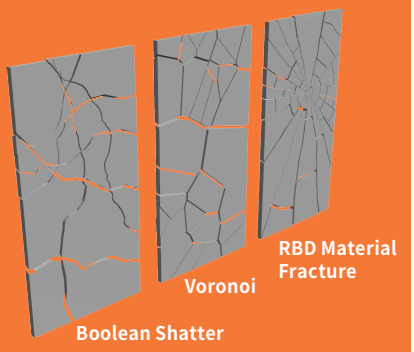
S を押して Select ツールを表示し、**4 を押して**プリミティブ選択に切り替えます。シートの 1 つを**ダブルクリック**して、シート全体を選択します。**Tab > Duplicate** を押して、コピーを作成します。**回転 (R)** ハンドルを使用して、複製したシートの向きを変更し、他のシートとほぼ直交させます。カップのみを切断し、ステムは避けるようにします。これにより、カップが別の方向に割れ、よりリアルなガラスの破片を作成できます。

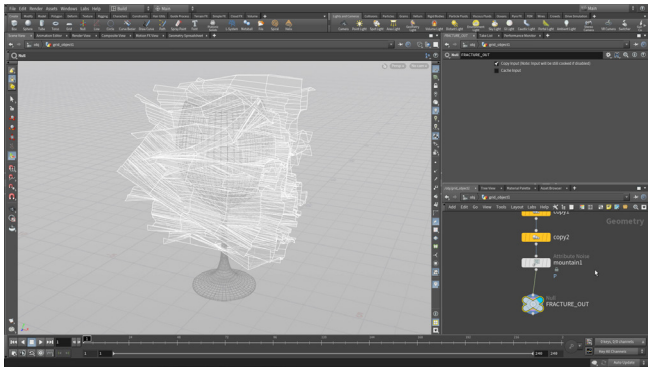
別のシートでも**繰り返し**、異なる角度で切断する別のサーフェスを作成します。

BOOLEAN SHATTER

Boolean ノードは、ほとんどの場合、**加算、交差、減算**といった従来のブーリアンを作成するのに使用されます。これらは閉じた形状にはうまく機能しますが、**Shatter** オプションを使用すればシートでジオメトリをスライスできます。

Houdini には **Voronoi Shatter** ツールもあり、使用することができますが、割れたガラスに必要なギザギザなルックにはなりません。また、ガラスのような粉砕を作成できる **RBD Material Fracture** ノードもありますが、これらは平坦なサーフェスでの使用に最適な機能なので、このワイングラスのレッスンでは使用していません。

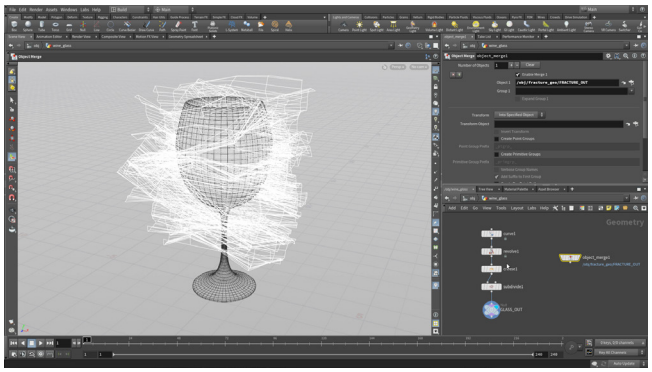




05 すべてのジオメトリを選択し、**Tab > Mountain** を押して、さまざまなシートのポイントにノイズを追加します。**Amplitude** を **0.75** に設定します。グラスの破砕がより面白いものになります。

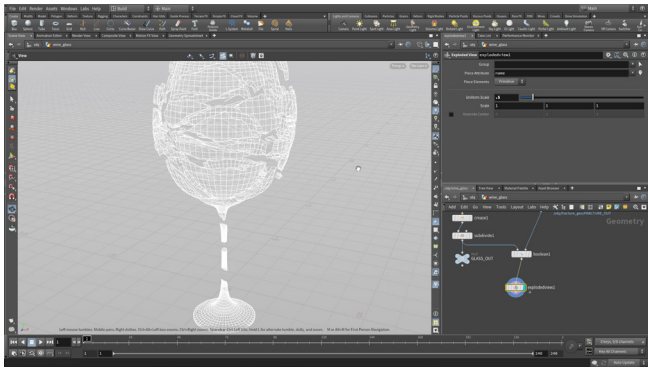
ネットワークビューで、**Null** ノードをチェーンの終端に追加し、その名前を ***FRACTURE_OUT*** に変更します。このノードに **Display フラグ**を設定します。

オブジェクトレベルに移動し、このノードの名前を ***fracture_geo*** に変更し、**Display フラグ**を**オフ**にして非表示にします。



06 ***wine_glass*** オブジェクトの中に入ります。ネットワークビューで、**Tab > Object Merge** を選択し、ノードを配置します。**Object 1** の横にあるノードセレクタをクリックし、**fracture_geo > FRACTURE_OUT** に移動して、このノードを選択します。**Transform** が **Into Specified Object** に設定されていることを確認してください。

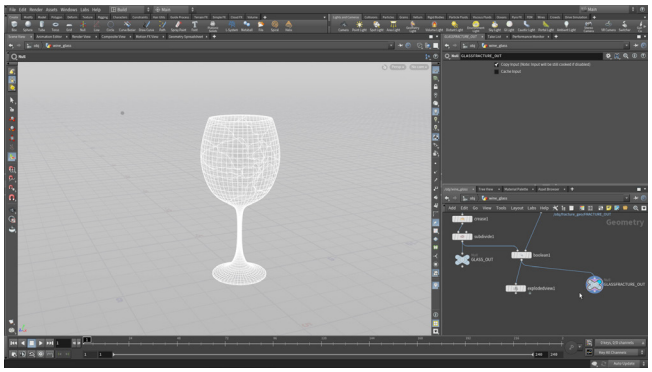
これにより、ノイズを含むシートがワイングラスジオメトリネットワークに追加され、グラスを Boolean で Shatter できるようになりました。



07 ネットワークビューで **Tab > Boolean** を選択し、クリックして新しいノードを配置します。***Subdivide*** ノード（***GLASS_OUT*** ではありません）を 1 つ目の入力に接続し、***object_merge*** を 2 つ目に接続します。その **Display フラグ**を設定したら、次のように設定します。

- **Set B: Treat As** を **Surface** にする
- **Operation** を **Shatter (Pieces of A)** にする

破砕を確認するため、**Exploded View** ノードをチェーンの終端に追加します。破砕する様子を変更したい場合は、***fracture_geo*** オブジェクトに戻ってシートを編集します。編集内容は、プロシージャルに更新されます。



08 ネットワークビューで、***exploded_view*** ノードをバイパスする **Null** ノードを追加し、その名前を ***GLASSFRACTURE_OUT*** に変更します。このノードに **Display フラグ**を設定します。

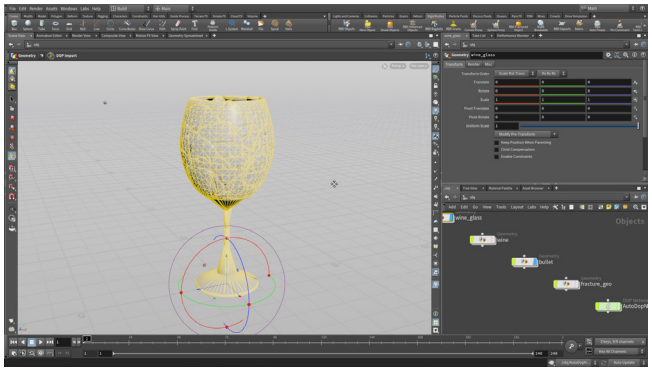
これで、このネットワークに 2 つの出力ノードができました。GLASS_OUT で元の形状のワイングラスが出力され、GLASSFRACTURE_OUT で破砕したグラスが出力されます。これら両方を流れに沿って使用し、ショットを完成させます。

作業内容を**保存**します。

パート 4

RBD シミュレーションのセットアップ

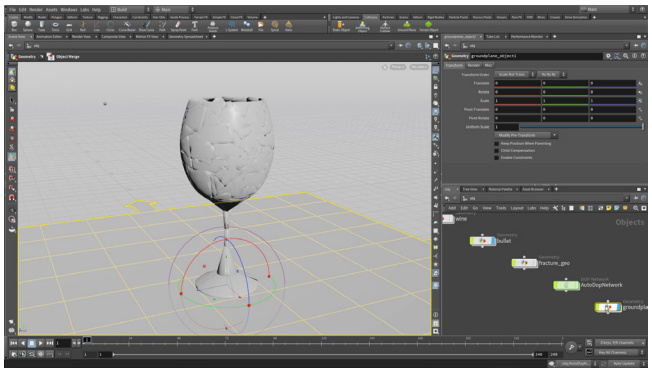
シェルフツールを使用してリジッドボディシミュレーションを作成します。これにより、ジオメトリ、フォース、ソルバノードをまとめた DOP（Dynamic Operator）ネットワークが追加されます。ワイングラスのジオメトリネットワークには、シミュレーション用のジオメトリ準備のためのノードが追加されます。Convex Proxy を使用して、Bullet RBD ソルバが不規則なガラス破片を処理できるようにします。



01

オブジェクトレベルで ***wine_glass*** オブジェクトを選択し、**Rigid Bodies** シェルフの **RBD Convex Proxy** ツールをクリックします。これにより、***AutoDopNetwork*** という初期ダイナミクスネットワークがセットアップされます。**V を押して Shading > Smooth Wire Shaded** を選択します。

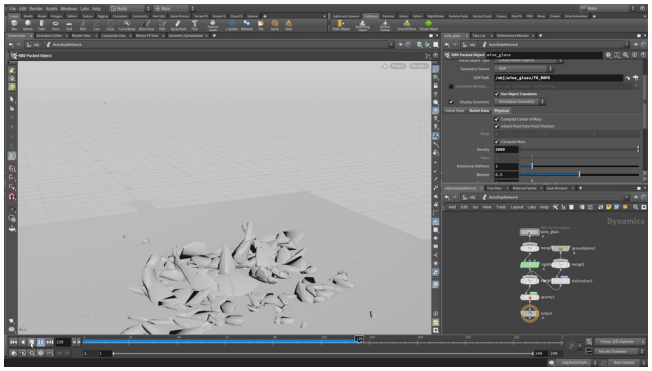
このツールを使用し、複雑な衝突の作成に使用できる凸状形状にワイングラスのパーツを分解します。ソースジオメトリよりも粗く表示されますが、シミュレーションの後で、クリーンなトポロジを確認できます。



02

Collisions タブに移動し、**Ground Plane** シェルフツールをクリックします。これにより、衝突するジオメトリ用に無限の地面が作成されます。

groundplane_object の **Display フラグ**を**オフ**にして、Scene View に表示されないようにします。それでもシミュレーションでは引き続き衝突するサーフェスとして機能します。



03

AutoDopNetwork に入ります。***wine_glass*** ノードを選択し、**Physical** タブで **Density** をガラスのおおよその密度である **2000** に設定します。

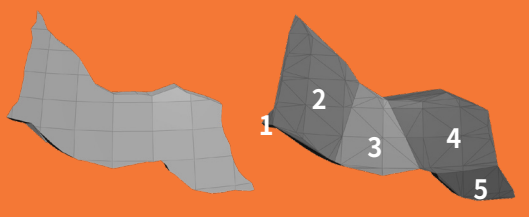
プレイバーで **Play** を押し、どうなるかを確認します。ガラスが地面に落下します。現時点では、破片に作用する力は重力しかありません。

衝撃まで破片を接着しておく **Glue ネットワーク**をセットアップしてもかまいませんが、弾丸は非常に高速なため、パーツの接着は不要です。

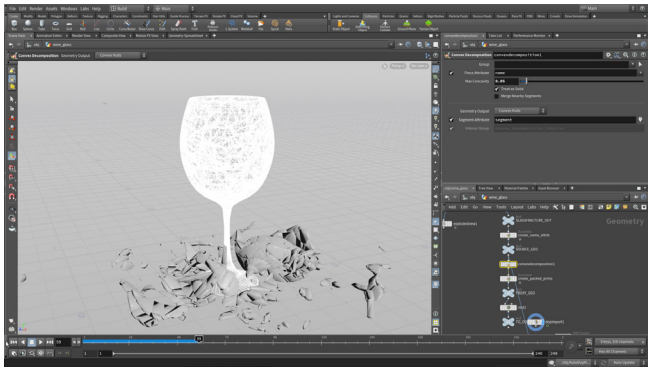
CONVEX DECOMPOSITION

RBD シミュレーションでは、高速シミュレーションを維持するために、Houdini は 凸型形状を好む Bullet ソルバを使用します。**Convex Decomposition** では、凹型の形状をつながった凸型の形状に分解できます。その後、これらは Bullet ソルバによって 1 つの合成ピースとしてシミュレートされます。

ワイングラスの破片の形状はさまざまなので、Convex Decomposition によってすべての破片が正確に衝突するようにします。



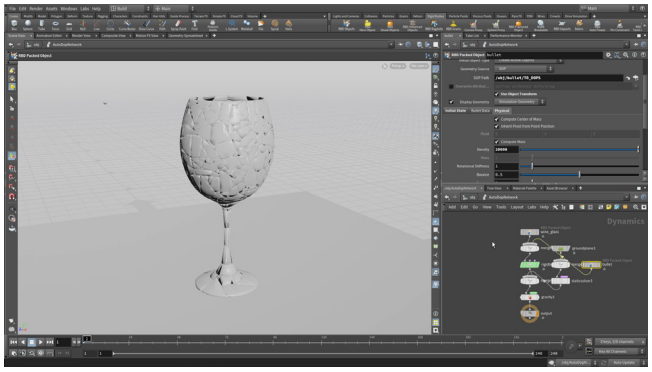
元のジオメトリ　　Convex Decomposition



04 オブジェクトレベルに上がって ***wine_glass*** オブジェクトに入ります。たくさんのノードを追加してプロキシジオメトリを作成しており、このチェーンは現在表示されている ***dopimport*** ノードで終わります。

convexdecomposition ノードで、**Max Concavity** を **0.05** に変更して衝突ジオメトリを調整します。

Play を押してシミュレーションを確認します。

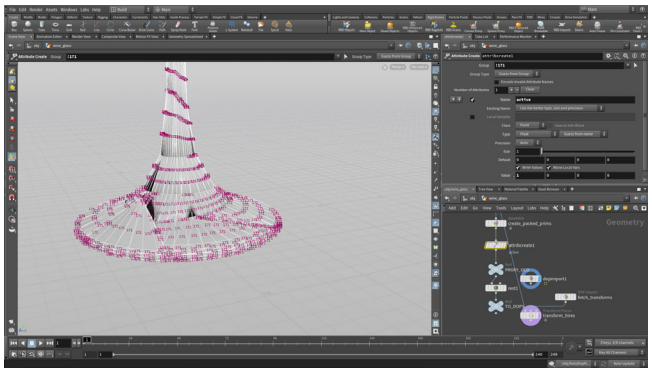


05 オブジェクトレベルに上がって ***bullet*** オブジェクトを選択し、**Rigid Bodies** シェルフの **RBD Objects** ツールをクリックします。弾丸からパックされた RBD オブジェクトが作成され、ダイナミクスネットワークに追加されます。

AutoDopNetwork に移動し、***bullet*** ノードを選択します。

- **Initial State** タブで、**Velocity** を **400, 0, 0** にする
- **Physical** タブで、**Density** を **20000** にする

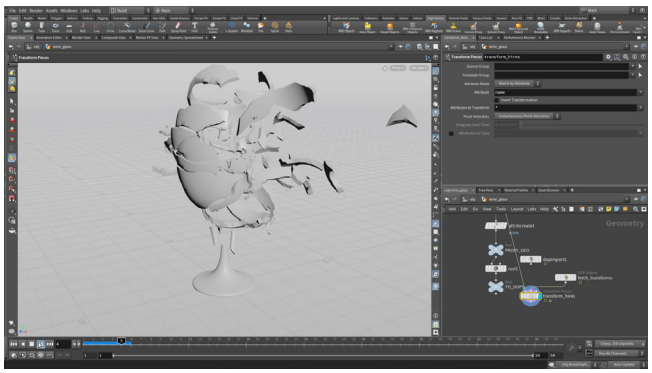
これは鉛の密度で、弾丸に適しているはずです。



06 シミュレートする前に、グラスのプレートが地面に固定されたままになるようにします。***wine_glass*** オブジェクトに移動して、ディスプレイバーで**プリミティブ**番号をオンにします。ここではプレートのパックプリミティブは 171 ですが、皆さんのものはおそらく違っているでしょう。

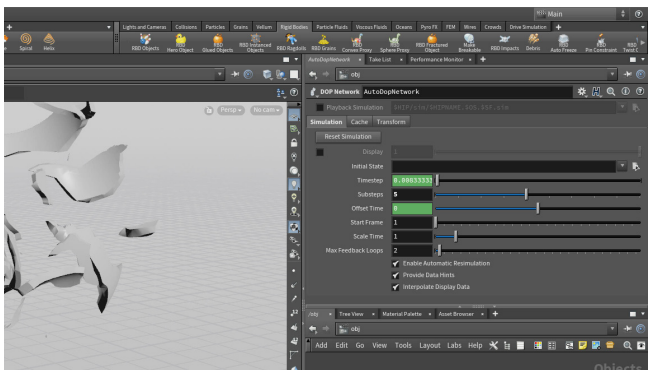
Attribute Create ノードを ***create_packed_primitive*** と ***proxy_geo*** ノードの間に作成します。次のように設定します。

- **Group** を **!171**（または皆さんのプレートの番号）にする
- **Name** を ***active*** にする
- **Value** を **1, 0, 0, 0** にする



07 次に、***transform_hires*** ノードの **Display フラグ**をオンにして、**Global Animation Options** を開き、**End** を **50** に設定します。

シミュレーションを**再生**します。ソースジオメトリがプロキシに合わせてアニメートされているのが分かりますが、衝突はそれほどドラマチックではありません。



08 ソルバでの弾丸シミュレーションでは、デフォルトの**サブステップ**は **10** です。これは、弾丸のスピードの衝突を解決するのに十分ではありません。オブジェクトレベルに上がり、***AutoDopNet*** ノードで **Substeps** を **5** に設定します。これで、ソルバ設定でサブステップが追加されます。これによりシミュレーション時間が長くなりますが、正確さが増すうえに、たいていはシミュレーションがよりアクティブになります。

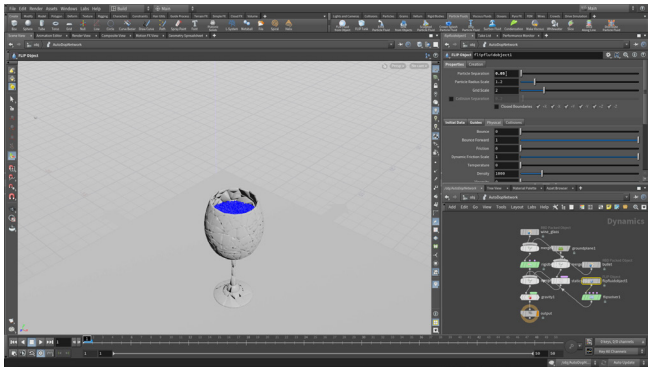
シミュレーションをもう一度**再生**して、衝突の様子が改善されていることを確認します。何か変更したい場合は、戻って設定を調整します。戻って一部の亀裂の位置を変更してもかまいません。

作業内容を**保存**します。

パート 5

シミュレーションへ流体を追加

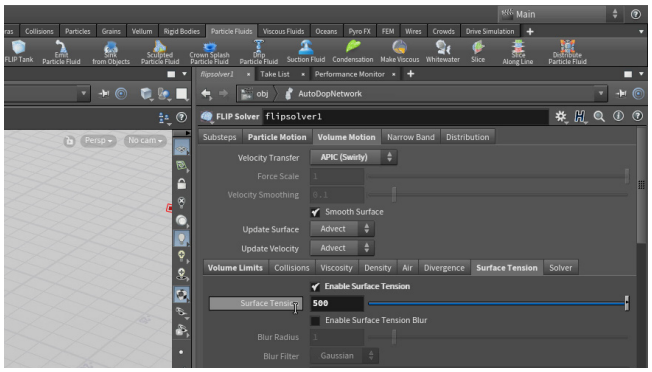
弾丸によるグラスの粉砕ができたので、ワインオブジェクトを流体に変換し、シミュレーションの一部として統合します。つまり、RBD と流体シミュレーションを同一 DOP ネットワークにまとめ、1 つのシステムとして動作するようにします。最初は、流体はパーティクルで表現されますが、サーフェス化して流体を視覚化できます。



01

フレーム 1 に進みます。ネットワークビューで ***wine*** オブジェクトを選択し、**Particle Fluid** シェルフの **FLIP Fluid from Object** ツールをクリックします。

すると、流体が流体パーティクル群になります。***AutoDopNet*** で、***flipfluidobject*** ノードを選択し、**Particle Separation** を **0.05** に設定します。これによりパーティクルが増え、シミュレーションにディテールが加わります。



02

flipsolver1 ノードを選択して **Particle Motion** タブをクリックし、**Behavior** で次のように設定します。

- **Add ID Attribute** を**オン**にする

Reseeding で次のように設定します。

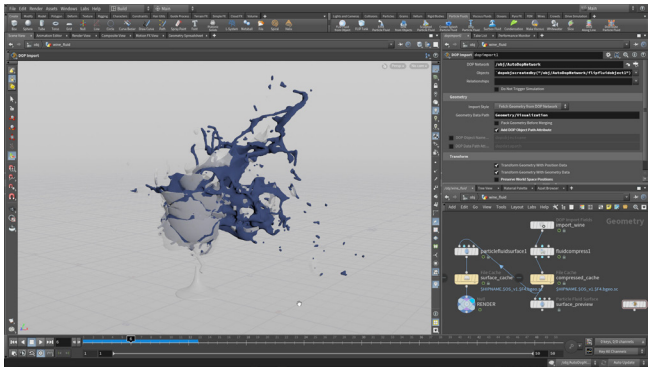
- **Reseed Particles** を**オフ**にする

Volume Motion で次のように設定します。

- **Velocity Transfer** を **(APIC) Swirly** にする

Surface Tension で次のように設定します。

- **Enable Surface Tension** を**オン**にする
- **Surface Tension** を **500** にする



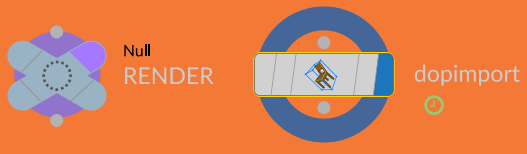
03

Play を押してシミュレーションを実行します。追加のサブステップのため時間は少し長くかかりますが、より正確な結果が得られます。**フレーム 10** を過ぎたら、**Escape** を押してシミュレーションを止めます。**フレーム 10** に移動して、ここまでの流体をプレビューします。

これをサーフェスとして表示するには、オブジェクトレベルに上がって ***wine_fluid*** オブジェクトに移動します。**Render** Null ノードに **Display フラグ**を設定すると、サーフェス化された流体を確認できます。これはクックするのに長い時間がかかりますが、サーフェスがどのように見えるかをフレーム毎に確認することができます。

Display フラグとレンダーフラグ

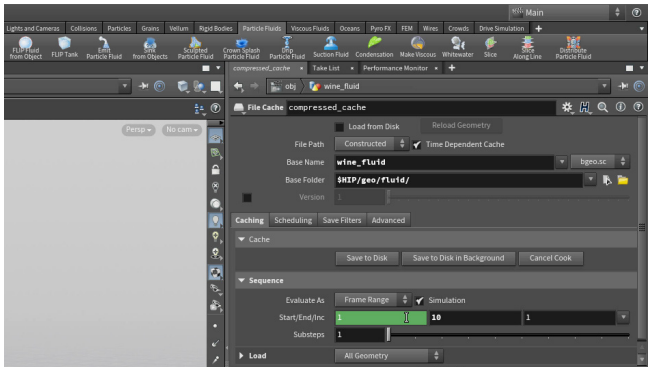
wine_fluid ジオメトリネットワークが最初に作成されたとき、**Display フラグ**はパーティクルを表示する **dopimport** ノードにありますが、**レンダーフラグ**は **particle surface** ノードから生じた Null ノードにあります。このセットアップにより、ビューポートでのパフォーマンスが高速になり、レンダリングした場合は最終的なサーフェスを確認できます。このレッスンでは、サーフェスをキャッシュ化するため、ノードはレンダリングに使用されません。



パート 6

シミュレーションのキャッシュ化とリタイム

このショットでは、10 フレームのシミュレーションを計算します。これをディスクに保存したら、Retime ノードで、流体が減速してから時間が逆戻りする長いショットに伸ばします。リタイムされた流体パーティクルがサーフェス化され、レンダリングのための最終ショットを定義する 50 フレームのシーケンスとして出力されます。その後、ワイングラスと弾丸も流体に合わせてリタイムします。



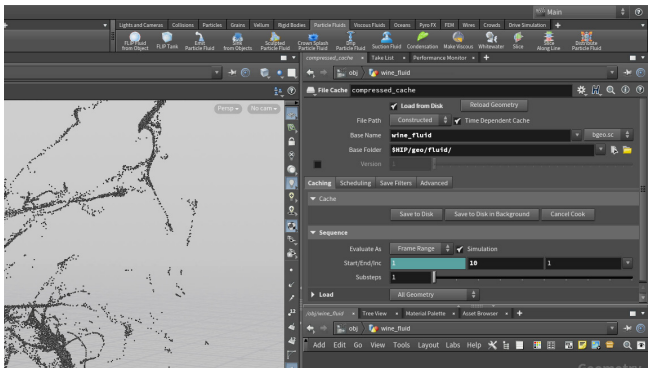
01

オブジェクトレベルで、シェルフツールで作成した ***wine_fluid_interior*** ノードを**削除**します。***wine_fluid*** オブジェクトに入って、***import_wine***、***compressedcache*** ノード、***particlefluidsurface*** ノードを除くすべてのノードを削除します。

compressed_cache で次のように設定します。

- **Base Name** を ***wine_fluid*** にする
- **Base Folder** を ***$HIP/geo/fluid/*** にする
- **Version** チェックボックスを**オフ**にする
- **End** を **10** にする（最初に **RMB クリック > Delete Channels**）

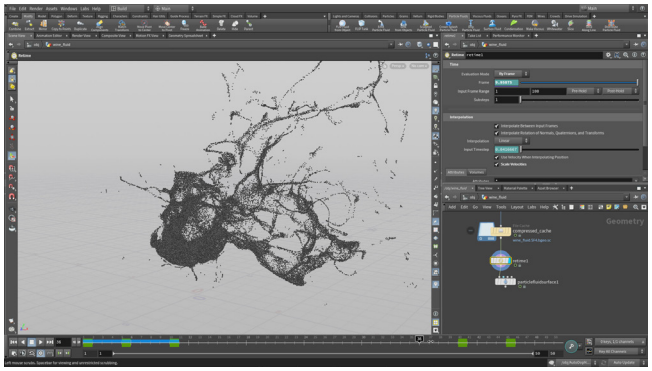
Save to Disk を押します。



02

完了したら、ジオメトリレベルのまま、Scene View の右上にある **Visibility** メニューから、**Hide Other Objects** を選択します。

Load from Disk を**オン**に設定して、ジオメトリシーケンスを**スクラブ**します。10 フレームのみ再生します。これからシーケンスをリタイムし、50 フレーム以上に伸ばしてエフェクトを減速します。



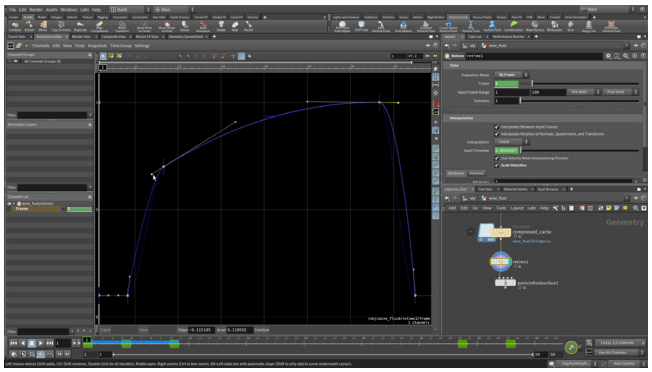
03

compressed_cache ノードの後に **Retime** ノードを追加して、**Display フラグ**を設定します。次のように設定します。

- **Evaluation Mode** を **By Frame** にする
- **Scale Velocities** オプションを**オン**にする

Frame フィールドで **RMB クリック**し、**Delete channels** を選択します。

- **フレーム 1**：**Frame** を **1** に設定し、**Alt クリック**してキーフレームを設定
- **フレーム 5**：**Frame** を **1** に設定してキーフレームを設定
- **フレーム 10**：**Frame** を **7** に設定してキーフレームを設定
- **フレーム 40**：**Frame** を **10** に設定してキーフレームを設定
- **フレーム 45**：**Frame** を **1** に設定してキーフレームを設定



04

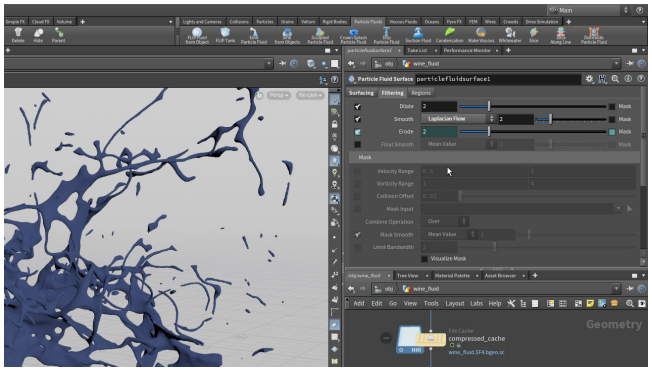
Animation Editor ペインをクリックして、作成したアニメーションカーブを確認します。グラフ上で **H を押して**ビューをホームし、**RMB クリックしながらドラッグ**して少しズームアウトします。

カーブハンドルを**クリックアンドドラッグ**し、この図と同じような形状にします。カーブの接線を分割する必要がある場合は、キーを選択し、**T を押して**接線を統一します。次に、それぞれの端を個別に選択してドラッグします。ここでの目標は、液体が素早く飛び出した後、ゆっくりになって短時間止まり、その後スピードアップして元の形状に素早く戻るようにすることです。

リタイム

ワイングラスの粉砕で一番面白い部分は、最初の 10 秒間です。シミュレーションのこの部分を強調するため、10 フレームの流体パーティクルを保存してから、**Retime** ノードを使用してシーケンスを伸ばします。時間を逆戻りさせて元のワイングラスに戻す、一種の「バレットタイム」効果を作成します。

これを流体に設定したら、ポイントをサーフェス化して、長いシーケンスを保存します。同じ Retime ノードは、粉砕するグラスと弾丸にコピーアンドペーストして使用できます。



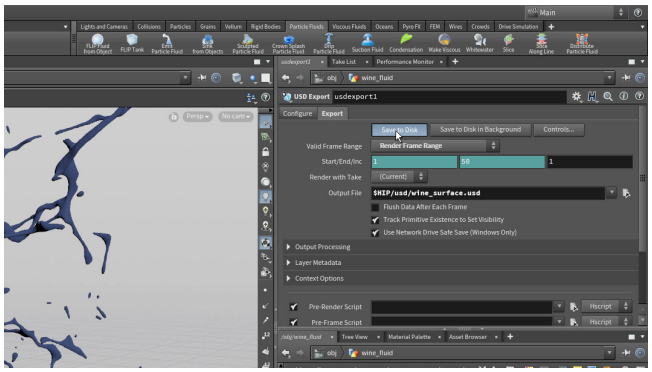
05

Retime ノードが ***particlefluidsurface*** ノードに接続されていることを確認し、Display フラグを設定します。これにより、リタイムをベースとした最終的な流体が得られます。***particlefluidsurface*** ノードを選択して次のように設定します。

- **Method** を **Average Position** にする
- **Union Compressed Fluid Surface** を**オフ**にする

Filtering タブで次のように設定します。

- **Dilate** を**オン**にし、**2** にする
- **Smooth** を**オン**にし、**Laplacian Flow** にする

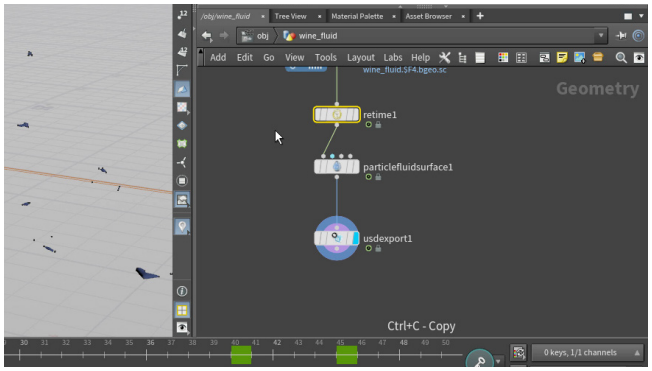


06

チェーンの終端に **USD Export** ノードを追加します。それを ***wine_surface*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/wine_surface.usd*** にする

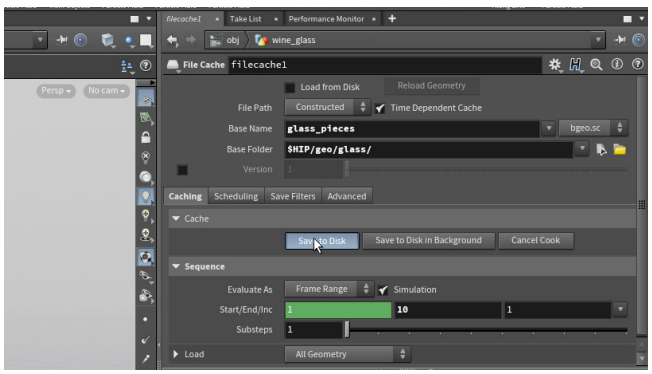
Save to Disk を押します。



07

Retime ノードを選択し、**Ctrl + C** を押してコピーします。別のネットワークでこれをペーストして、砕けるワイングラスをリタイムします。これにより、両方のネットワークでキーフレームが一致します。

作業内容を**保存**します。



08

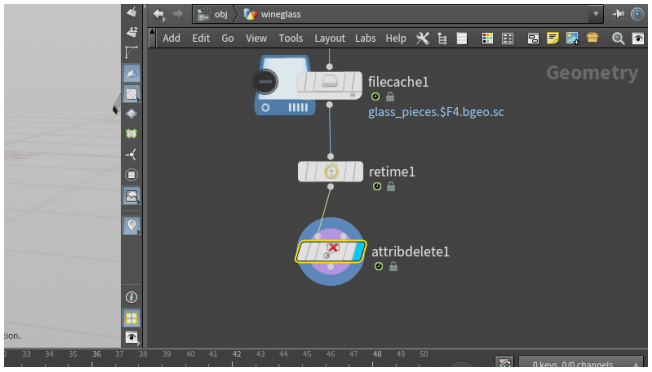
フレーム 1 に行き、***wine_glass*** ネットワークに移動します。***transform_hires*** ノードの下に、**File Cache** ノードを配置して接続し、次のように設定します。

- **Base Name** を ***glass_pieces*** にする
- **Base Folder** を ***$HIP/geo/*** にする
- **Version** チェックボックスを**オフ**にする
- **End** を **10** にする（最初に **RMB クリック > Delete Channels**）

Save to Disk を押します。**Load from Disk** を**オン**に設定したら、フレーム 1 から 10 までを**スクラブ**して、正確に見えることを確認します。

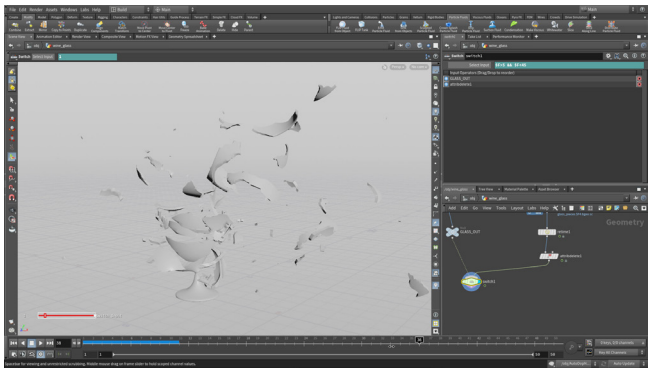
USD と SOLARIS

このプロジェクトのルックデベロップメント工程をサポートするため、**流体**、**ワイングラス**、**弾丸**を USD にキャッシュ化します。こうすることで、シミュレーションの再計算を気にすることなく、レンダリングに集中できるようになります。ここではシミュレーションと同じシーンファイルにキャッシュを表示しますが、新しいシーンファイルにキャッシュをインポートしてもかまいません。こうするとショットのライティングとレンダリングに集中できますが、シミュレーションに戻って微調整するのが大変になります。



09 **Ctrl + V を押して *Retime*** ノードをペーストし、それを ***filecache*** ノードの後に配置して **Display フラグ**を設定します。これで流体とグラスでタイミングが一致しました。タイムラインをスクラブして、破片のタイミングを確認します。

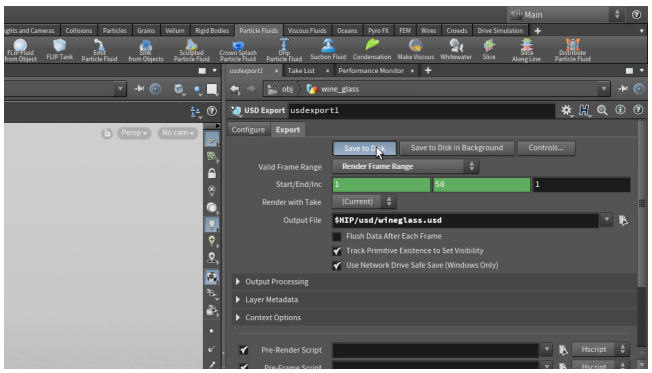
Retime ノードの後に、**Attribute Delete** を追加します。**Primitive Attributes** の横にある矢印から、***name*** を選択して、ジオメトリからこのアトリビュートを削除します。



10 **Switch** ノードをネットワークに追加します。***GLASS_OUT***、それから ***attribdelete*** ノードをそれに接続します。

Select Input を ***$F>5 && $F<45*** に設定します。タイムラインでスクラブして、その動作を確認します。このエクスプレッションは、フレーム 5 でひび割れのないグラスから割れたグラスに切り替わり、フレーム 45 で元に戻るようにするものです。

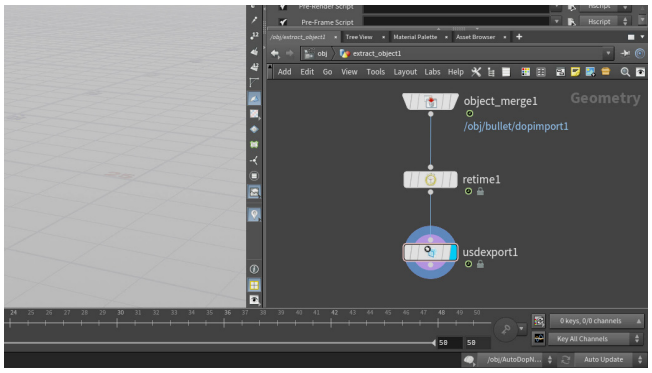
これらの形状をまとめて、ワイングラスの USD ファイルで、衝撃の前後はグラスが割れていないようにします。



11 チェーンの終端に **USD Export** ノードを追加します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/wineglass.usd*** にする

Save to Disk を押します。



12 **フレーム 1** に移動し、***bullet*** オブジェクトを選択します。**Modify** メニューから **Extract** を選択します。これにより、ジオメトリのワールド空間位置を取得できます。***extract_object*** の中に入り、***retime*** ノードを ***object_merge*** ノードの下に**ペースト**して、それらを接続します。

チェーンの終端に **USD Export** ノードを追加します。それを ***bullet*** という**名前に変更**します。次のように設定します。

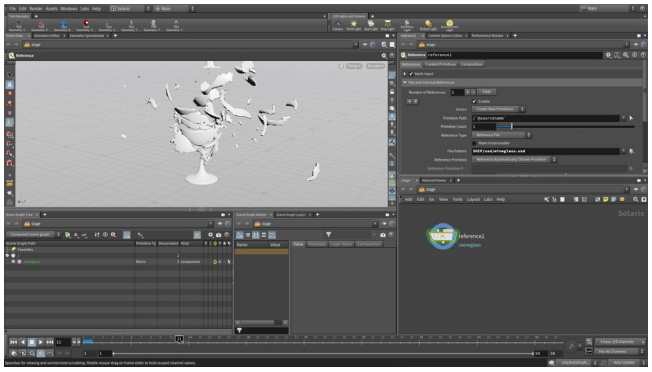
- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/bullet.usd*** にする

Save to Disk を押します。

パート 7

ショットの設定とレンダリング

ショットのレンダリングには、USD ファイルを Solaris ステージで参照し、背景を追加します。Solaris は Houdini のコンテキストで、LOP ノードを使用して USD シーングラフを定義します。次に、カメラを配置し、環境ライトを追加します。その後、ビューポートで Karma レンダラを実行し、ショットのプレビューレンダリングを行います。

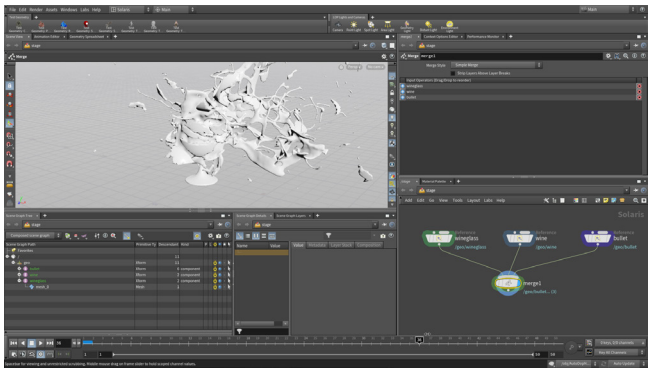


01

デスクトップを **Solaris** に変更します。パスバーで **Stage** を選択します。

ネットワークビューで **Tab > Reference** を選択してからクリックし、**Reference** ノードを追加します。**Reference File** の横にある **File Chooser** をクリックして、***wineglass.usd*** ファイルを指定します。ノードの名前を ***wineglass*** に変更します。**Primitive Path** を ***/geo/$OS*** に設定し、ノード名を使用して、***geo*** というグループに配置されるようにします。

Scene View で、ビューをホームする**スペースバー + H** のような表示ツールを使用して、ワイングラスがよく見えるようにします。



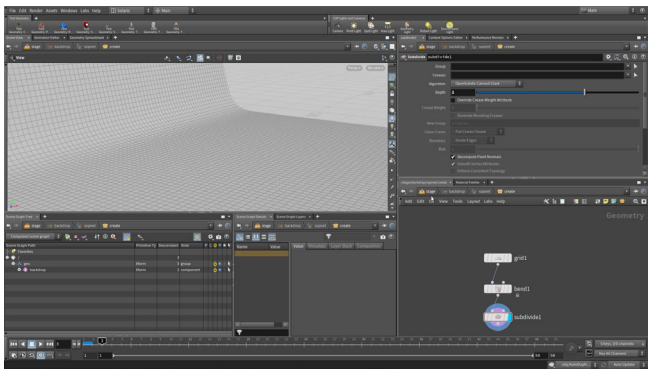
02

このノードを **Alt ドラッグ**して複製し、さらにもう 1 つ複製します。1 つ目のコピーで、**File Chooser** をクリックして ***wine_surface.usd*** ファイルを指定します。ノードの名前を ***wine*** に変更します。

2 つ目のコピーでも同じことを行い、***bullet.usd*** ファイルを指定します。ノードの名前を ***bullet*** に変更します。

ネットワークに **Merge** ノードを追加して、3 つすべての参照ノードをそれに接続します。**Display フラグ**を設定したら、スクラブして結果を確認します。

シーングラフには、***geo*** エントリの下に 3 つの参照 USD ファイルが表示されているはずです。



03

ネットワークビューで、**Tab** を押して **Grid** と入力します。クリックしてノードを配置したら、名前を ***backdrop*** に変更して、***Merge*** ノードに接続します。**Import Path Prefix** を ***/geo/$OS*** に設定します。***backdrop*** ノードを**ダブルクリック**して、ジオメトリレベルに入ります。

Grid ノードを選択し、**Size** を **200, 200**、**Rows** および **Columns** を **20** に設定します。***Grid*** ノードの出力を **RMB クリック**して、**Bend** と入力します。クリックして Bend ノードを配置したら、**Display フラグ**を設定します。**Bend** を **75**、**Capture Origin** を **-40**、**0**、**0**、**Capture Direction** を **-1**、**0**、**0**、そして **Capture Length** を **20** に設定します。***Grid*** ノードの出力を **RMB クリック**して、**Subdivide** と入力します。**Display フラグ**を設定し、**Depth** を **2** に設定します。

シミュレーションのキャッシュ化

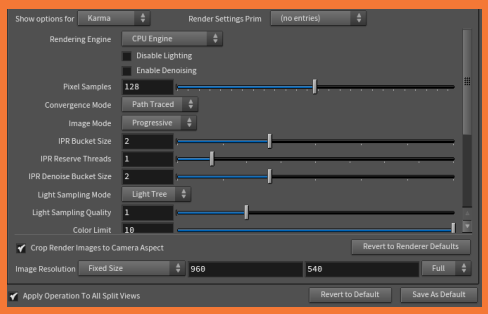
このプロジェクトのルックデベロップメント工程をサポートするため、**流体**、**ワイングラス**、**弾丸**をジオメトリ（USD）シーケンスにキャッシュ化しました。こうすることで、シミュレーションの再計算を気にすることなく、レンダリングに集中できるようになります。

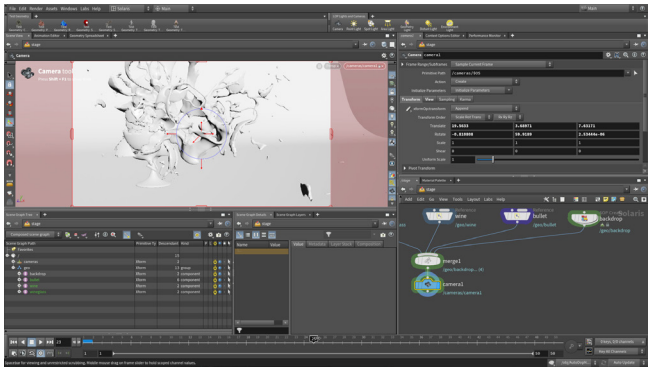
大量のハードディスク容量を消費する VFX ショットでは、こうしたワークフローが一般的です。たくさんのパーティクルを含む巨大なシミュレーションを送る際は、この点に注意するようにしてください。さまざまな中間段階を保管できる場所を確保することが重要です。

ビューポートレンダリング

Houdini のレンダラである Karma を使用して、シーケンスをレンダリングします。はじめは、Display Options の設定でレンダリングを実行します。Display Options を表示するには、Scene View で **D** を押します。デノイザをオンにしたり、Pixel Samples や Image Resolution の設定が可能です。

その後、**Karma LOP** をセットアップすれば、最終出力はそのノードのレンダリング設定で実行されます。





04 表示ツールを使用して、正面から ***wineglass*** が見えるようにします。**LOP Lights and Camera** シェルフで、**Camera** ツールを **Ctrl クリック**します。ネットワークに Camera ノードが加わり、カメラ越しにビューポートを見られるようになります。

Lock Camera/Light to View ボタンを押し、ビュー変更に応じてカメラの位置が更新されるようにします。ビューポートで**タンブル**、**パン**、**ドリー**してカメラを再配置し、ワイングラスが左側にきて、しぶきは右側に飛ぶようにします。タイムラインをスクラブし、シーケンス全体でカメラが機能していることを確認します。

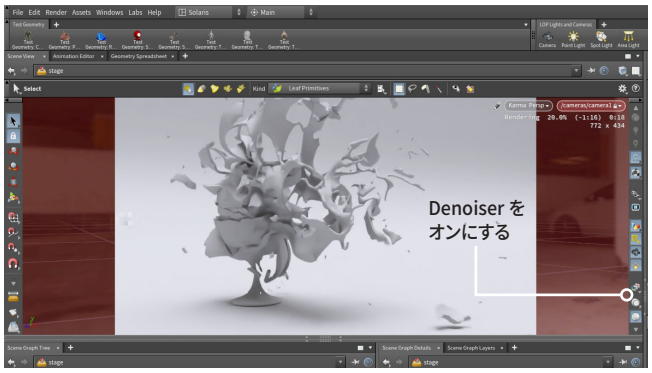
05 **LOP Lights and Camera** シェルフで、**Camera** ツールを **Ctrl クリック**します。***Domelight*** ノードをチェーンの終端に追加します。

Domelight ノードを選択し、**Base Properties** タブで **Texture** の横にある **File Chooser** ボタンをクリックします。サイドバーで ***$HFS/houdini/pic/hdri*** リストをクリックして、***HDRIHaven_skylit_garage_2k.rat*** ファイルを選択します。**Accept** をクリックします。

Display Options バーで、**High Quality Lighting with Shadows** ボタンをクリックします。

06 ***Persp*** メニューで **Karma** を選択し、ビューポートで Karma を使ってレンダリングします。タイムラインで異なるフレームに移動すると、ビューポートが素早く更新されます。

Karma は USD を使用するよう設計されているので、LOP コンテキストのすべてが USD シーングラフに変換されます。Houdini のこの部分からのみ Karma レンダラを使用できます。

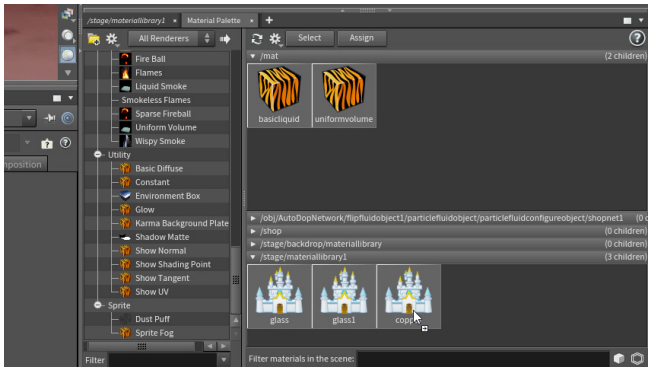


07 レンダリング時によりクリーンな画像を得るには、Nvidia グラフィックカードがあり、最新のドライバをインストールしている場合は **Denoiser** をオンにします。Denoiser は、**Display Options** でオンにできます。

パート 8

マテリアルの割り当てとシーケンスのレンダリング

最後に、ワイングラス、ワイン、弾丸にマテリアルを追加します。これらのマテリアルは USD シーングラフの一部となり、LOP ノードでジオメトリに割り当てられます。次に、Karma LOP で、Nvidia Optix Denoiser を含むレンダリングの設定を行い、レンダリング後、シーケンスを MPlay にロードし、結果を確認します。

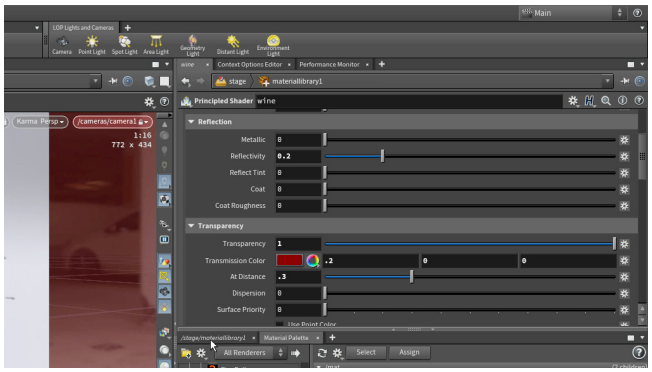


01

ネットワークビューで **Tab > Material Library** を押します。それをチェーンの終端に接続し、**Display フラグ**を設定します。

Material Palette ペインに移動します。***/stage/materiallibrary*** の横にある矢印をクリックし、このエリアを開きます。パレットの左側のマテリアルギャラリーをスクロールして、ワイングラスとワイン用に **Glass** マテリアルを 2 つ ***materiallibrary*** 作業エリアにドラッグします。

次に **Copper** マテリアルを見つけ、***materiallibrary*** 作業エリアにドラッグします。これは弾丸に使用します。

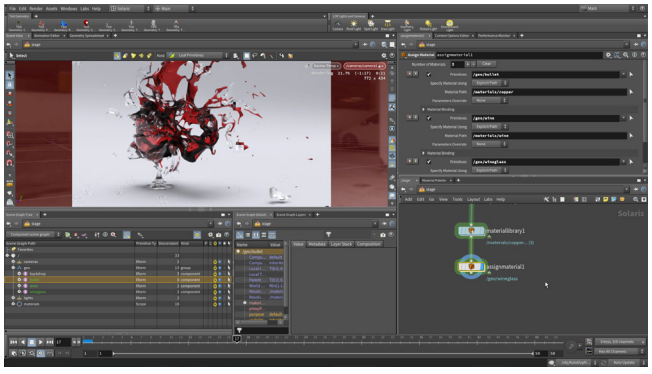


02

2 つ目の ***Glass*** マテリアルを選択し、名前を ***wine*** に変更します。

ワインの IOR である **Inside IOR** を **1.3443** に設定します。
次に、**Reflectivity** を **0.2** に設定し、環境の反射率を下げます。

Transmission Color を **0.2, 0, 0** に設定して、赤みがかったワインの色に変更します。それから **At Distance** を **0.3** に設定します。



03

ステージレベルに戻ります。Material Library の後に **Assign Material** ノードを追加します。

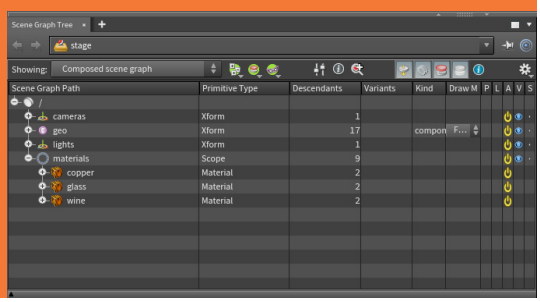
シーングラフから ***wineglass*** を **Primitives** フィールドにドラッグしたら、**Material Path** の横にある矢印をクリックして、このプリミティブ用に ***glass*** マテリアルを選択します。

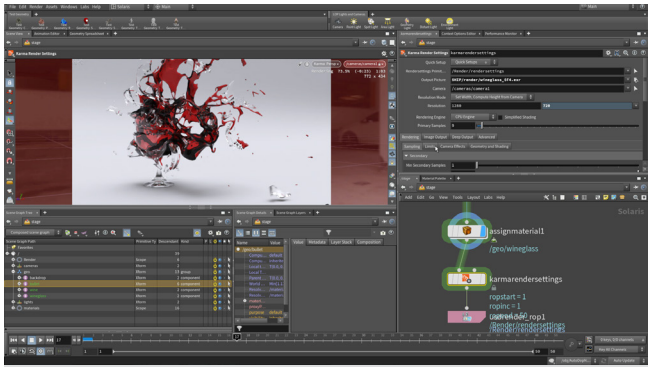
次にチェックボックスの横にある **+(プラス)記号**を 2 回クリックして、2 つの新しいエントリを追加します。同じ方法で ***wine*** マテリアルを ***wine*** プリミティブに割り当て、***copper*** マテリアルを ***bullet*** プリミティブに割り当てます。

シーングラフ

ジオメトリやライトと同じように、LOP ノードを使用して追加したマテリアルは**シーングラフ**に追加されます。Material Library LOP を使用したとき、Material Path Prefix のデフォルト設定は **/materials/** でしたが、それはグラフでのマテリアルの位置でもあります。編成を変えることも可能ですが、これがデフォルトです。

このマテリアルパスは、マテリアルをジオメトリに配置するために **Assign Materials LOP** で使用したパスです。

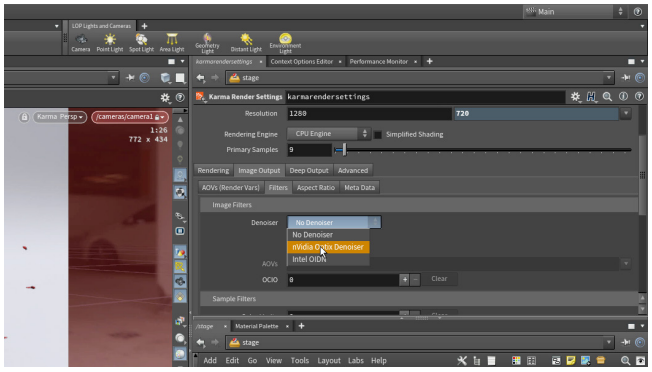




04

ネットワークビューで、**Tab > Karma** を押し、**Karma Render Settings** と **USD Render ROP** ノードを追加します。それらをチェーンの終端に接続します。***karmarendersettings*** ノードを選択して、**Image Output > Filters** タブで **Denoiser** を **nvidia Optix Denoiser** に設定してデノイザをオンに戻します。

usdrender_rop ノードを選択します。**Valid Frame Range** を **Render Frame Range** に設定し、**Output Picture** を ***$HIP/render/wineglass_$F4.exr*** に設定します。名前の ***$F*** は、レンダリングにフレーム番号を付加するのに必要で、***4*** はフレーム番号のパディングです。

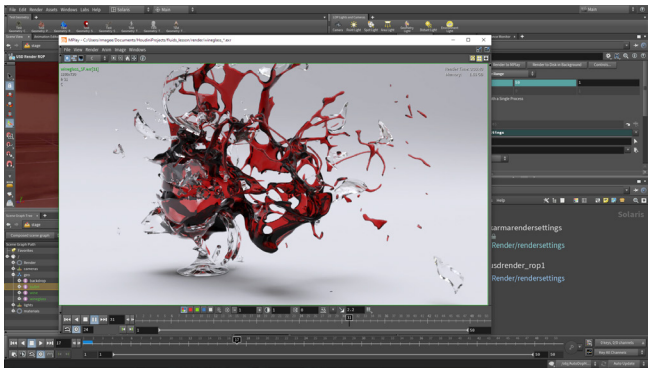


05

ビューポートの Denoiser はこのノードからの出力に影響しないので、明示的に選択します。***karmarendersettings*** ノードを選択して、**Image Output > Filters** タブで **Denoiser** を **nvidia Optix Denoiser** に設定してデノイザをオンに戻します。

nVidia Optix Denoiser がビューポートで使用されるものと一致します。**Intel OIDN** デノイザもありますが、ディスクにレンダリングする場合のみ利用可能です。

作業内容を**保存**します。***usdrender_rop*** ノードを選択して、**Render to Disk** をクリックします。



06

完了したら、**Render > Mplay > Load Disk Files** を選択し、レンダリングした画像を開いて最終的なシーケンスを確認します。

後で別の **Karma** ノードを分岐させて、解像度とレンダリング設定を上げて最終的なレンダリングを行います。**Convergence Mode** に戻って **Variance** に設定し、**サンプル**数を上げ、**デノイザ**をオフにします。最初は低解像度でテストレンダリングを行い、すべてが希望通りになっていることを確認するようにしてください。

まとめ

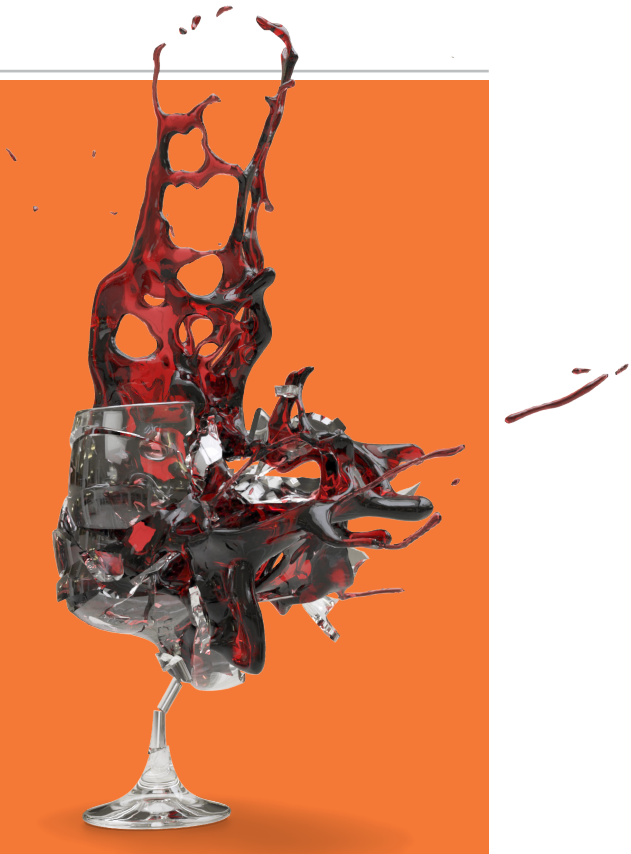
Bullet RBD と FLIP 流体ソルバを使用して、ワイングラスを粉砕する VFX ショットを完成させました。Retime ノードを使用して、速度を落としてから時間を逆戻りさせ、ワイングラスが最初と同じ位置に戻るようにしました。そして結果を USD ファイルにキャッシュ化しました。

その後、Solaris/LOPS コンテキストを使ってショットを設定し、ライトを追加しました。マテリアルを作成してプリミティブに割り当て、ショットに適切な外観を得ることができました。

このプロジェクトでは、Houdini のダイナミクスノードとネットワークを使用してさまざまな種類のエフェクトを統合したり、ジオメトリノードを使用してシミュレーションのセットアップと出力を行う方法を紹介しています。

VFX ショットの作成に使用されるノードとネットワークを理解できたら、Houdini をさらに深く探求し、独自のエフェクトを作成してください。

楽しむことが一番です！



HOUDINI FOUNDATIONS

破壊 FX

ビジュアルエフェクトが楽しい理由の 1 つは、現実には何も破壊せずに、さまざまなものを爆破できることです。このレッスンでは、パーティクルの火花を使用して導火線に火を付け、カートゥーンの爆弾を爆発させます。爆弾にはリジッドボディダイナミクス、火と煙には Pyro FX を使用します。このレッスンでは、さまざまなシェルフツールやネットワークノードを使用して、ダイナミックシミュレーションをセットアップする方法を学びます。

作成するショットを完全に理解するために、すべての要素をゼロから作成して、エフェクトをシミュレートします。Houdini シーンのさまざまなコンテキストで、シミュレーションノードがどう機能するかを理解しやすくなるはずです。最後に、Karma レンダラを使用してショットをレンダリングします。

ACES | OPENCOLORIO セットアップ

Pyro FX の使用時、より正確にカラーを表示するには、**Academy Color Encoding System (ACES)** を使用します。そのためには、Scene View の**ビューポート (Persp)** メニューから、**Correction ツールバー**を表示します。右にある矢印ボタンから、**OpenColorIO** を選択します。これにより、ディスプレイが **sRGB**、出力が **SDR Video - ACES 1.0** になります。この設定は現在のセッションにのみ有効で、Houdini を開くたびに戻す必要があります。

レッスンの目標

爆弾をモデリングし、パーティクルの火花、リジッドボディダイナミクス、Pyro FX を使用して爆発させます。

学習内容

- **爆弾**をモデリングし、**導火線**をアニメートする方法
- **カメラをアニメート**し、ショットをセットアップする方法
- **すすの軌道 (トレイル)** と導火線の**火花**をセットアップする方法
- 爆弾のジオメトリを**粉砕**して爆発させる方法
- 爆発用の **Pyro FX** シミュレーションをセットアップする方法
- **マテリアル**と**テクスチャ**をセットアップする方法
- **Solaris** コンテキストで **Karma** を使用してエフェクトを**レンダリング**する方法

使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

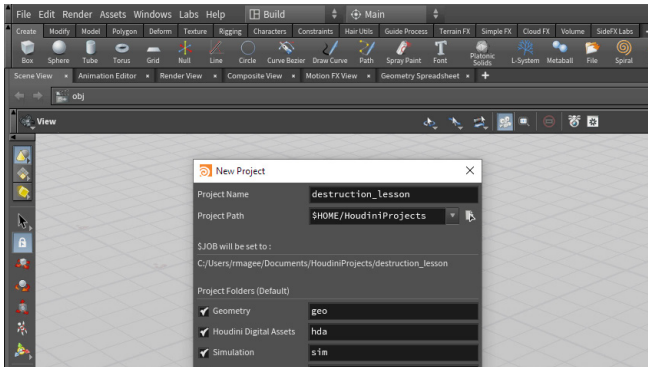
このレッスンの手順は、以下の Houdini 製品で実行可能です。

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

ドキュメントバージョン 4.0.1J | 2023 年 8 月

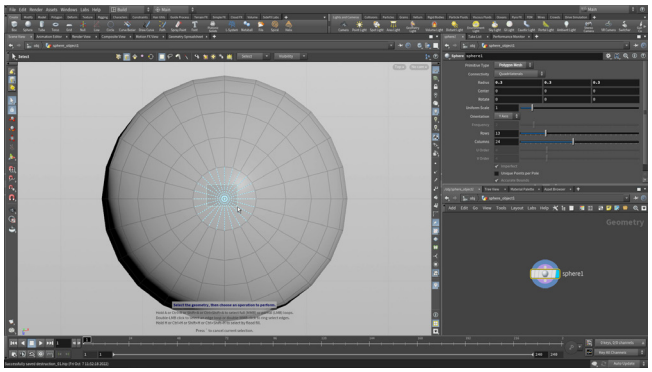
パート 1
爆弾のモデリング

爆弾のジオメトリを作成するには、プリミティブの球から始め、上部を開口部に変更します。
これにはポリゴンの押し出しとベベルを使用して、最終形状に必要なジオメトリを定義します。
レッスンの後半で、この爆弾を砕きます。



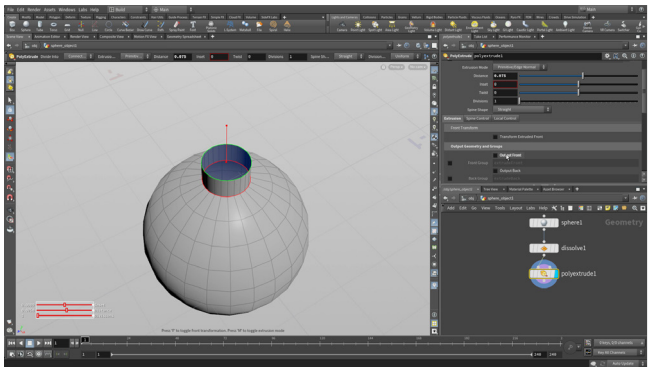
01 **File > New Project** を選択します。**Project Name** を ***destruction_lesson*** に変更し、**Accept** を押します。これにより、プロジェクトディレクトリとサブディレクトリが作成され、このショットに関連するすべてのファイルが格納されるようになります。

File > Save As... を選択すると、新しい ***destruction_lesson*** ディレクトリが表示されます。ファイル名を ***destruction_01.hip*** に設定し、**Accept** をクリックして保存します。



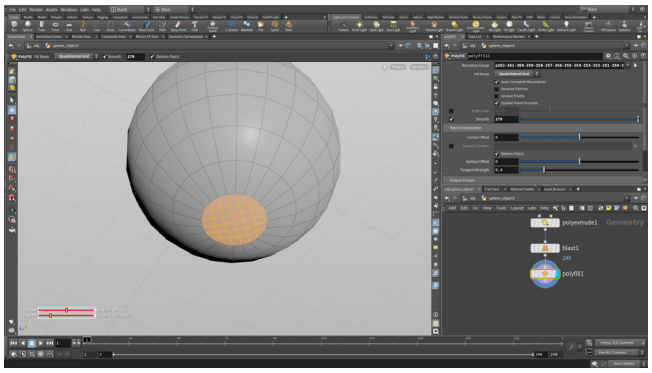
02 ビューポートで、**C を押して** Radial メニューを表示します。このメニューから **Create > Geometry > Sphere** を選択します。ビューポートで、**Enter** を押して球を原点に配置します。上部のオペレーションコントロールツールバーで、**Radius** を **0.3, 0.3, 0.3** に設定します。

S を押して Select ツールにしたら、**3** を押してエッジ選択を呼び出します。**スペースバー + 2** を押して Top ビューに移動します。球の上部と下部のエッジを矩形選択し、**Delete** を押します。これによりエッジが削除され、2 つの円形ポリゴンができます。**スペースバー + 1** を押してパースビューに戻ります。



03 **S を押して** Select ツールにし、**4** を押してプリミティブ（フェース）選択にします。球の上部の円形ポリゴンを選択します。

C を押して Radial メニューを表示し、**Model > Polygons > PolyExtrude** を選択します。**Distance** を約 **0.075** にして、ハンドルを上に移動します。**Output Front** を**オフ**にします。



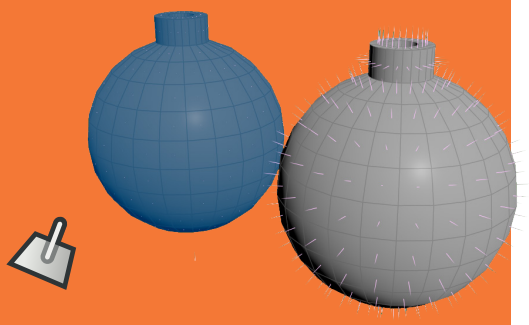
04 球の下部の円形ポリゴンを**選択**し、**Delete を押します**。これにより、ネットワークに **blast** ノードが追加されます。**S を押して Select** ツールに移動し、**3** でエッジ選択に切り替えます。先ほど作成した穴のエッジを**ダブルクリック**して、すべてのエッジを選択します。

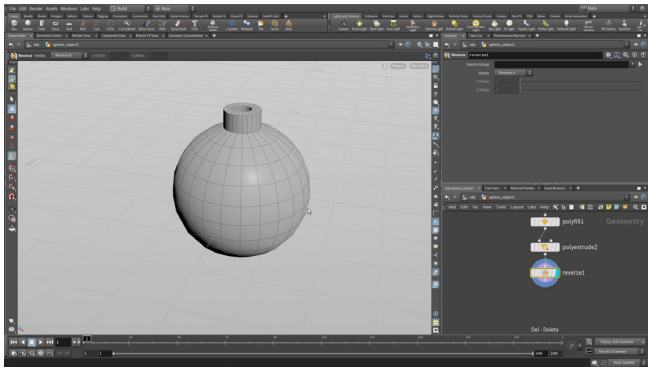
Tab > Polyfill を押します。パラメータエディタで、**Fill Mode** を **Quadrilateral Grid**、**Smooth** を **270** に設定します。球の下部に、単一のポイントに集中しない、クリーンなトポロジが作成されます。

サーフェス法線

すべてのプリミティブに法線方向があり、片方の側が内側、もう片方の側が外側になっています。爆弾のジオメトリを押し出すと、最初はジオメトリが裏返しになります。これは、青色のフェースで示されます。その後、Reverse ノードを使用して法線の方向を変えます。

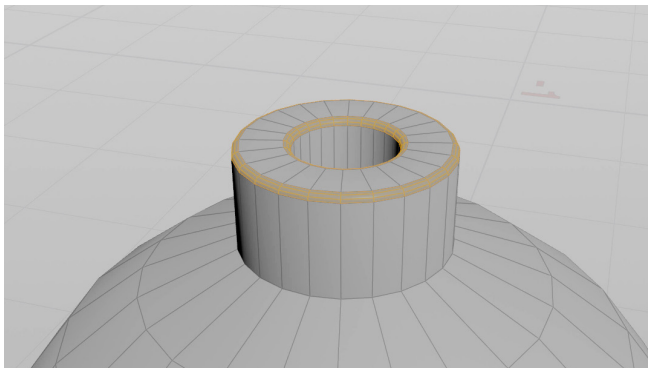
サーフェス上に法線を表示するには、**Scene View** ペインの右側の **Display Options** バーにある **Primitive Normals** ボタンを使用します。





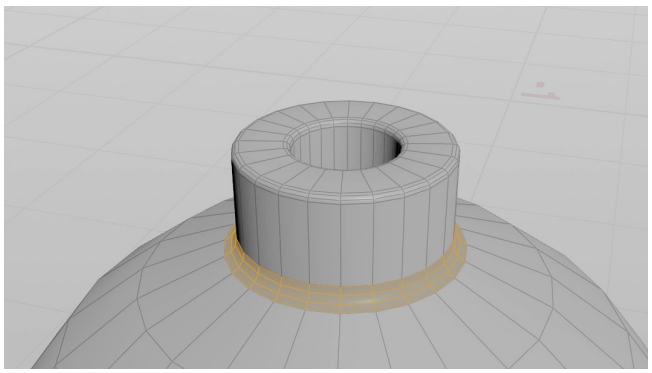
05 **N を押して**すべてのフェースを選択し、再度 **Polyextrude** ツールを選択します。約 **-0.04** の **Distance** 値に押し出します。パラメータエディタの **Extrusion** タブで、**Output Back** をオンにします。

N を押してすべてのフェースを選択し、再度 **Tab** を押して **Reverse** と入力していきます。このノードは、すべてのポリゴンの法線を反転させます。これで、法線が正しい方向を向くようになりました。



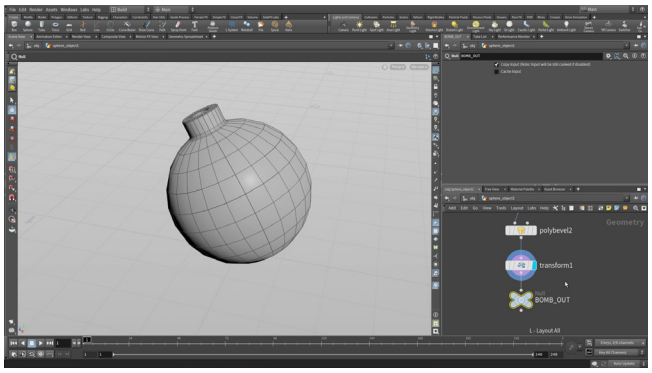
06 **S を押して Select** ツールに移動したら、**3** でエッジ選択に変更します。爆弾の上部のエッジを**ダブルクリック**したら、**Shift** を押しながら**ダブルクリック**して上部の内側の円を選択します。

C を押し、Radial メニューから **Model > Polygons > Polybevel** を選択します。**Distance** を **0.005** に設定します。**Shape** を **Round**、**Divisions** を **3** に設定します。



07 **S を押して Select** ツールに切り替えます。爆弾の円の部分と上部の押し出された部分が交わる箇所のエッジを**ダブルクリック**します。

Q を押して最後に使用した **Polybevel** ツールに戻り、**Distance** を **0.01**、Shape を **Round**、**Divisions** を **3** に設定します。



08 ネットワークビューで、**Tab > Transform** を押して、それをネットワークの終端に追加します。**Translate Y** を **0.3**、**Rotate X** を **27** 度に設定します。

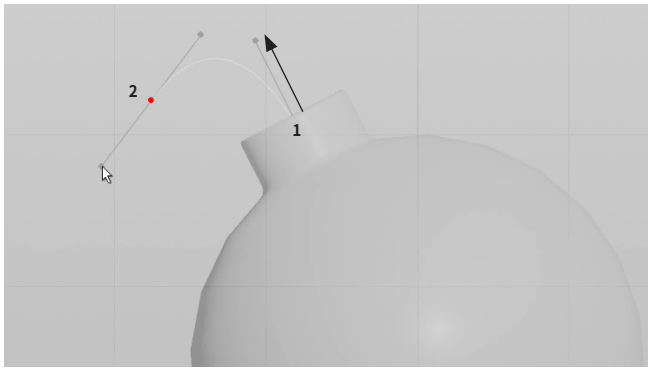
Null ノードを追加します。終端に ***null*** を接続し、***null*** ノードに **Display フラグ**を設定して表示します。null ノードの名前を**ダブルクリック**して、***BOMB_OUT*** に変更します。

オブジェクトレベルに移動して、オブジェクトの名前を ***bomb_geo*** に変更します（爆弾のジオメトリが含まれるため）。

パート 2

導火線のモデリング

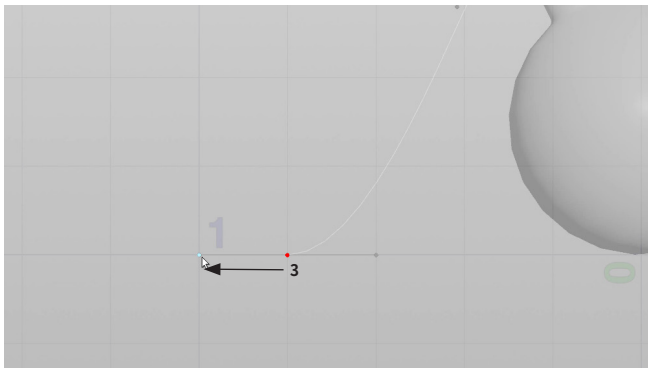
導火線の作成は、爆弾の上部から伸びる Bezier カーブから始めます。地面にカーブをくねらせて、長い導火線を作成します。カーブの方向を逆にして導火線のアニメーションの準備をしてから、Polywire ノードを追加して導火線に厚みを与えます。



01

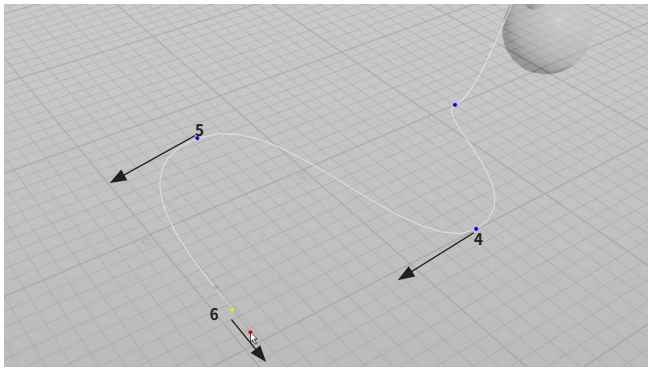
スペースバー + B を押してすべての角度から爆弾を表示します。**Right** ビューに移動し、**スペースバー + B** を押して拡大します。

C を押して Radial メニューを表示し、**Create > Geometry > Curve** を選択します。**クリックアンドドラッグ**して、カーブの 1 つ目のポイントと接線ハンドルを作成します。次に、ポイントを追加し、**下にドラッグ**して下向きのカーブを簡単に描画します。



02

X を押して Grid を選択し、グリッドスナッピングをオンにします。地面でクリックアンドドラッグして 3 つ目のポイントを作成し、接線ハンドルは地面に揃えます。



03

スペースバー + B を押して 4 面ビューに戻ります。マウスを**パース**ビューの上に移動し、再度**スペースバー + B** を押してビューを拡大します。

グリッドスナッピングをオフにし、**Display Options** バーの上から 2 つ目のボタンを使用して、**コンストラクション平面**をオンにします。これにより、カーブが地面から離れなくなります。

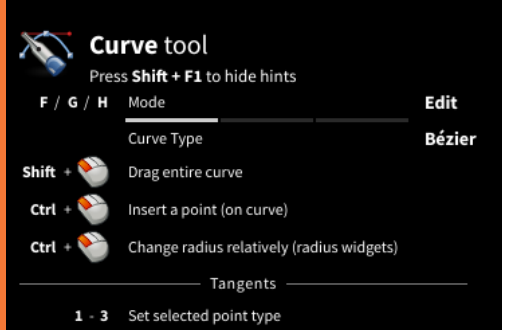
接線をドラッグして新しい 3 つのポイントを描画し、カーブの形状を定義します。

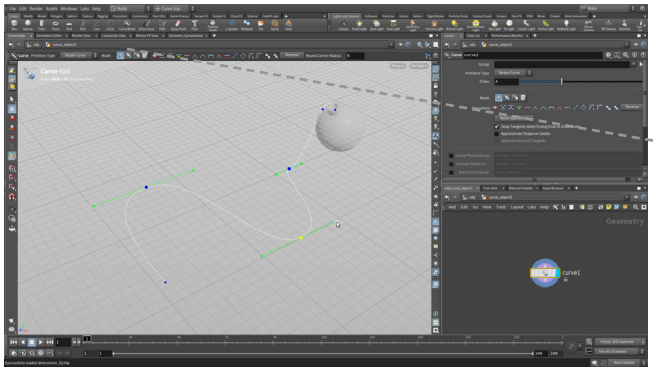
MMB クリックしてカーブを完成させます。

ツールのヒント

Curve ツールにはツールのヒントが付属しており、作業時に Scene View に表示されます。このツールのさまざまなショートキーオプションが表示され、ツールの機能を理解するのに役立ちます。

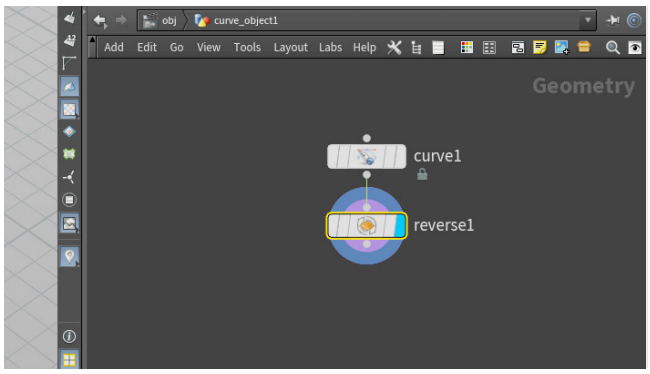
Shift + F1 を使用して折り畳むと、ツール名のみが表示されます。ヒント付きのツールは増えています。Houdini の今後のバージョンでは、さらに多くのツールでヒントが表示されるでしょう。



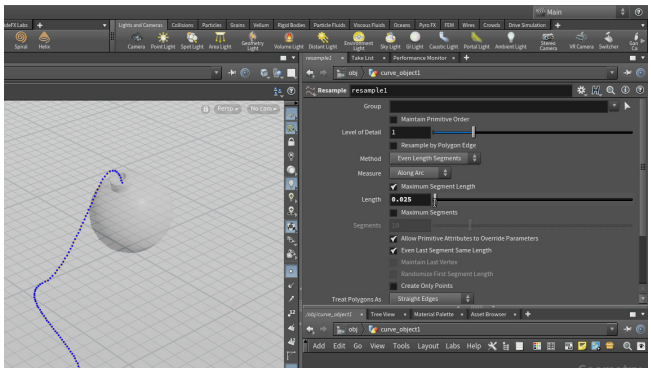


04 **オペレーションコントロール**ツールバーで、カーブの **Mode** を **Edit** に変更します。カーブ上の編集ポイントをクリックして、カーブの形状を微調整できるようになります。

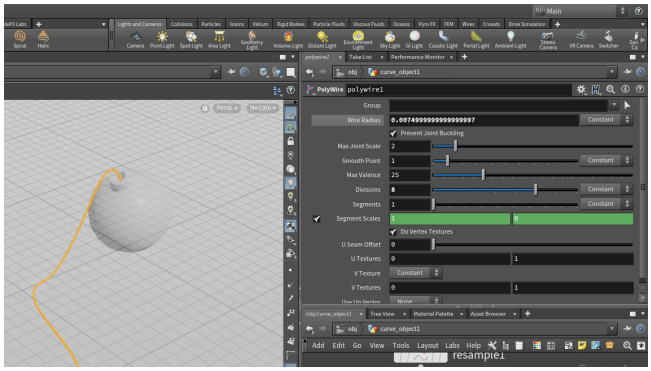
地面でポイントを選択して編集し、カーブを希望通りのルックにします。タンブルして、カーブが地面上にあることを確認します。



05 Scene View で、**Tab > Reverse** を押します。**N を押して**カーブ全体を選択し、**Enter** を押します。カーブを爆弾から出てきたように描いたため、このままでは逆の方向からアニメートされてしまいます。これで、導火線の火が付く側から、カーブが開始するようになります。



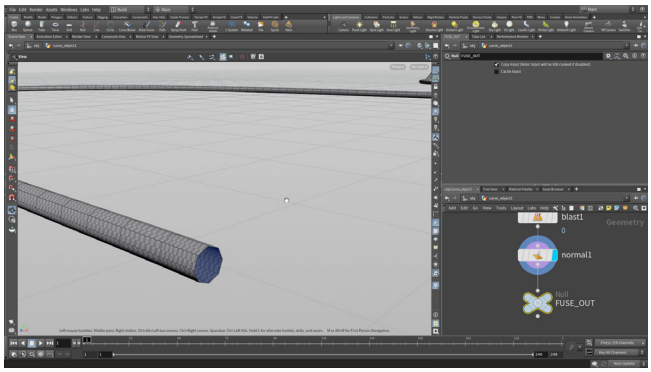
06 **Display Options** バーの **Points** オプションをオンにします。**Resample** ノードを追加します。***resample*** ノードで、**Maximum Segment Length** を **0.025** に設定して、ディテールを追加します。***resample*** ノードによって、ポイントが均一に配置されます。



07 **Polywire** ノードを追加して、ワイヤーに厚みを与えます。**Wire Radius** を **0.0075**、**Divisions** を **8** に設定します。

reverse ノードと ***polywire*** ノードの間に、**Transform** ノードを追加します。polywire ノードに移動し、**Wire Radius** パラメータを **RMB クリック**して、**Copy Parameter** を選択します。***transform*** ノードに戻り、**Translate Y** を **RMB クリック**して、**Paste Relative References** を選択します。

これで、導火線全体が持ち上がり、地面のグリッドに半分埋まっていた状態が解消します。



08 **Blast** ノードを追加して、**Group** を **0** に設定します。これにより、導火線のジオメトリの終端が削除されます。**Normal** ノードをチェーンの終端に追加します。

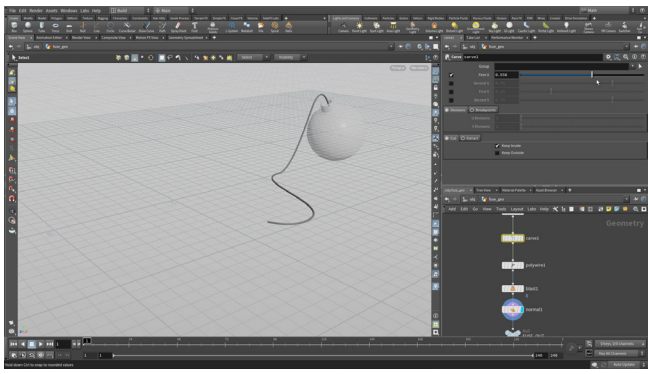
normal ノードの後に **Null** ノードを追加して、***FUSE_OUT*** と名前を付けます。これで、導火線全体のジオメトリを表すノードができました。

オブジェクトレベルに戻り、オブジェクトの名前を ***fuse_geo*** に変更します。

パート 3

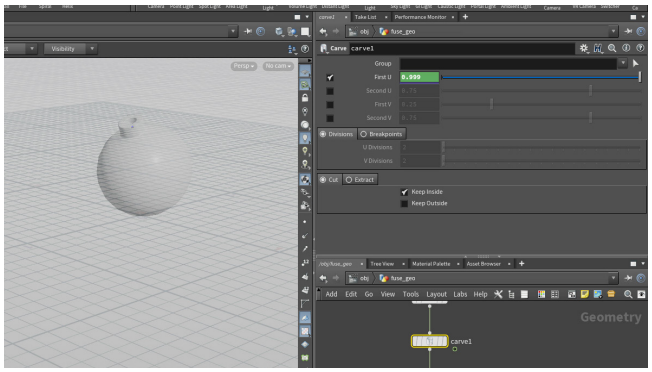
導火線のアニメート

導火線のアニメーションでは、Carve ノードを使用して、時間の経過とともにカーブの長さが変化するようにします。導火線に丸いキャップを追加して、すすと火花を放出するのに使用します。カーブ上に接線をセットアップして、キャップが適切に動くようにします。次に、NULL オブジェクトをいくつか追加して、パーティクルの放出に使用するキャップを簡単にエクスポートできるようにします。



01 ***fuse_geo*** オブジェクトの中に入ります。***transform*** ノードと ***Polywire*** ノードの間に、**Carve** ノードを追加します。**First U** スライドをドラッグして、カーブにどのような影響を与えるのか確認します。

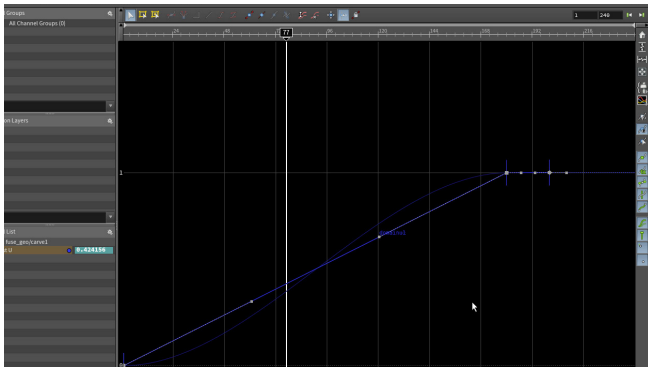
First U を **0** に設定します。**First U** を **Alt クリック**して、**フレーム 1** にキーフレームを設定します。パラメータボックスの色が変わり、キーフレームが設定されたことと、現在のフレームにキーがあることが示されます。



02 **フレーム 180** に移動します。**First U** を **0.999** に設定します。これにより、キーフレームが設定されます。**フレーム 200** に移動します。**First U** を **1.0** に設定します。これにより、もう 1 つキーフレームが設定されます。

プレイバーの左下で、再生が速くなりすぎないように **Real Time Playback** をオンに切り替えて、**Play** を押します。

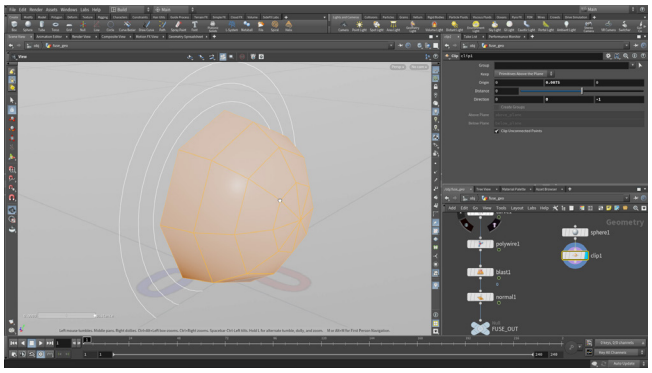
これで、導火線が爆弾のジオメトリに向かってアニメートするようになりました。爆弾には爆発をセットアップします。



03 **Animation Editor** ペインタブをクリックします。アニメーションカーブを選択して、パネル上部の **Straight** ボタンをクリックします。カーブが直線になり、導火線の最初から最後まで、速くなったり遅くなったりせず、均一の速度にアニメートされます。

Scene View ペインタブに戻って、アニメーションを**再生**し、この変更がモーションに与える影響を確認します。

作業内容を**保存**します。



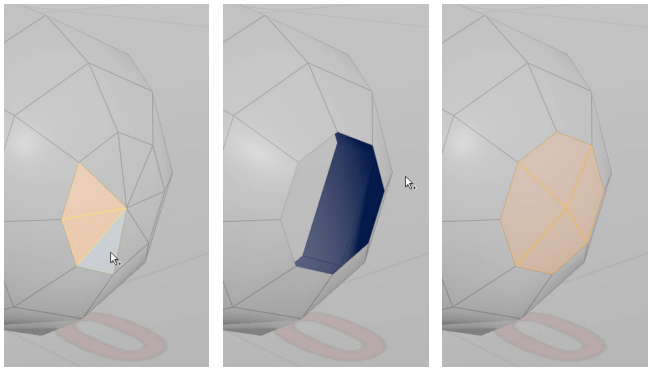
04 ネットワークビューで、**Sphere** ノードを追加して、**Display フラグ**を設定します。次のように設定します。

- **Radius** を **1, 1, 1** にする
- **Center Y** を **0.0075** にする
- **Uniform Scale** を **0.0075** にする

スペースバー + F を押して、球にフォーカスします。

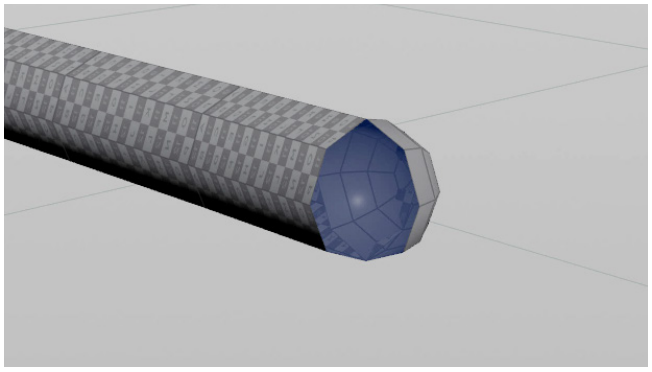
- **Orientation** を **Z Axis** にする
- **Rows** を **9**、**Columns** を **8** にする

Scene View で、**N を押して**すべてを選択してから、**Tab > Clip** ノードを選択し、**Direction** を **0, 0, -1** に設定します。



05 タンブルして、**S を押して** Select ツールにし、**4** を押して**フェース/プリミティブ**選択にします。球の先端で三角形の 1 つを選択したら、**A キーを押しながら** 2 つの三角形を **MMB クリック**して、三角形の面をすべて選択します。**Delete** キーを押して、それらを削除します。これにより、ネットワークに ***Blast*** ノードが追加されます。

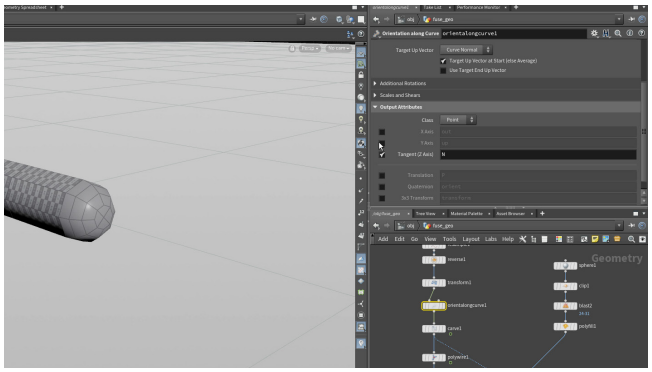
3 を押してエッジ選択に切り替え、先ほど削除した領域のエッジをダブルクリックします。**Tab > polyfill** を押して、***blast*** の後にノードを配置します。**Fill Mode** を **Quadrilateral Grid** に設定し、**Display フラグ**をオンにします。**Smooth** を **100** に、**Tangent Strength** を **0** に設定します。これにより、球の先端に四角形トポロジが作成されます。



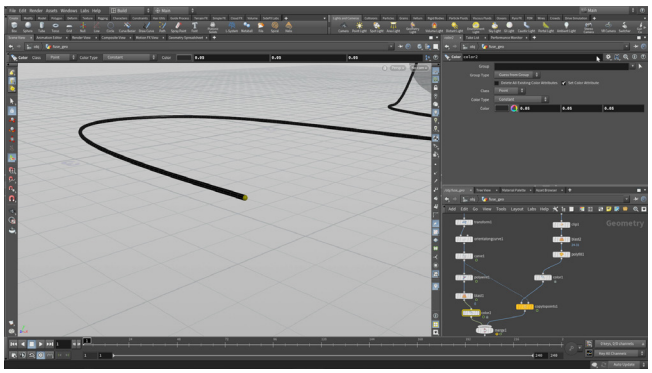
06 **Copy to Points** ノードをネットワークに追加します。球からの ***polyfill*** ノードを 1 つ目の入力に、***carve*** ノードを 2 つ目の入力に接続します。**Target Points** を **0** に設定します。

Merge ノードを追加します。***blast*** ノードと ***copytopoints*** ノードを接続してから、***normal*** ノードに接続します。

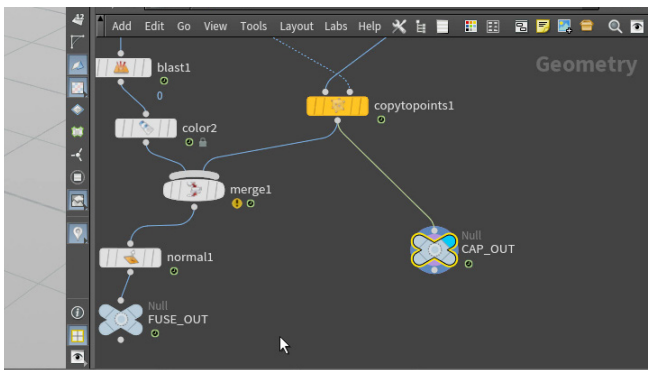
キャップはカーブの終端に適切に配置されましたが、方向が正しくありません。カーブに法線を追加して、適切に揃える必要があります。



07 ***reverse*** ノードと ***carve*** ノードの間に、**Orientation Along Curve** ノードを追加します。**Output Attributes** で **Y Axis** オプションをオフにします。**Tangent (Z Axis)** は **N** のままにしておきます。これにより、法線がカーブに追加され、先端のキャップが導火線に合わせて動くようになります。



08 ***polyfill*** ノードの後に **Color** ノードを追加して、Color を**黄色**に設定します。導火線の blast ノードの後に **Color** ノードをもう 1 つ追加して、**Color** を**ダークグレー**に設定します。こうした色分けは、作業中に、導火線を視覚化するのに役立ちます。また、これらの色を基に、後で割り当てるマテリアルに影響を与えることも可能です。



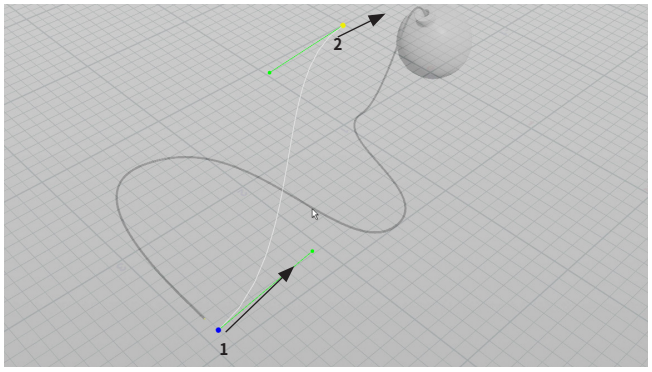
09 ***copytopoints*** ノードの後で **Null** ノードを分岐させて、横に配置します。***null*** ノードの名前を ***CAP_OUT*** に変更します。後でこれを使用して、キャップを別のネットワークに抽出します。パーティクルの放出では、そのネットワークを参照することになります。**Display フラグ**を設定し、球の半分が表示されることを確認します。**プレイバー**をスクラブして、球が導火線とともに動くことを確認します。

完了したら、**Display フラグ** の設定を FUSE_OUT null に戻します。

パート 4

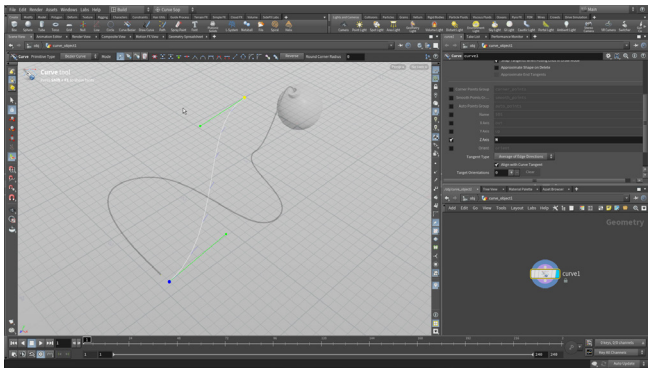
アニメーションカメラの作成

このショットをさらに発展させていくうえで、最終ショットをフレーミングするカメラをセットアップしておくと便利です。
このカメラリグを構築するには、Null オブジェクトをカーブに拘束してから、Aim 拘束を使用してカメラを Null オブジェクトに向けます。
カメラが導火線の終端に追従するようになるため、パーティクルの放出を評価しやすくなります。

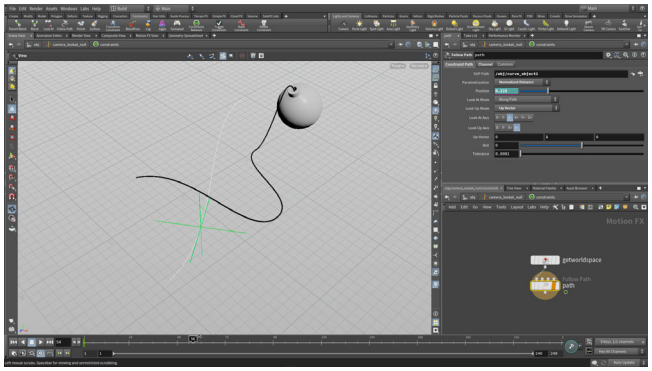


01 ドリーアウトして上からシーン全体を表示します。**コンストラクション平面**が**オン**になっていることを確認します。**C を押して** Radial メニューを表示し、**Create > Geometry > Curve** を選択します。導火線の開始付近のポイントを**クリックアンドドラッグ**したら、さらにドラッグして接線を伸ばします。

次に、爆弾の背後で 2 つ目のポイントを**クリックアンドドラッグ**したら、さらにドラッグして接線を伸ばし、カーブの形状を作成します。**MMB クリック**して終了し、形状を調整したい場合は **Edit** モードを使用します。

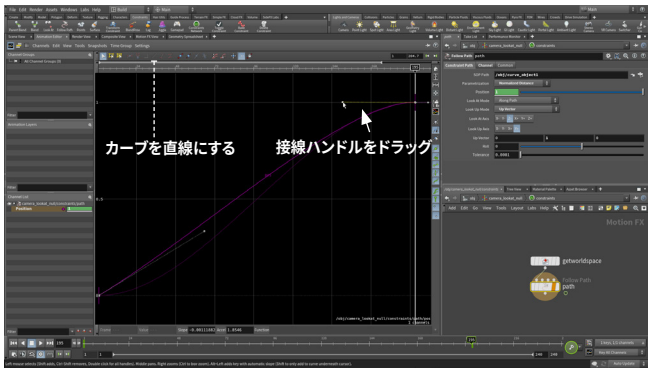


02 Curve ノードで、Z Axis オプションをオンにして、N に設定します。これにより、カーブに沿ってアニメーションを補助する法線が作成されます。



03 原点に **Null** オブジェクトを追加します。このノードの名前を ***camera_lookat_null*** に変更します。**Constraints** シェルフから、**Follow Path** ツールをクリックします。これで、Null が開始オブジェクトとして受け入れられます。パスオブジェクトとしてカーブを選択し、**Enter** を押します。**look at** オブジェクトまたは **look up** オブジェクトは必要ないので、**Enter** をもう 2 回押します。

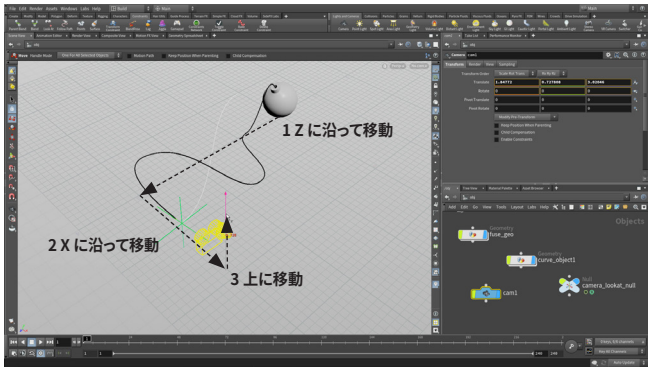
タイムラインをスクラブすると、開始フレームから終了フレームまで、Null オブジェクトがパスに沿って一定のペースで移動することを確認できます。



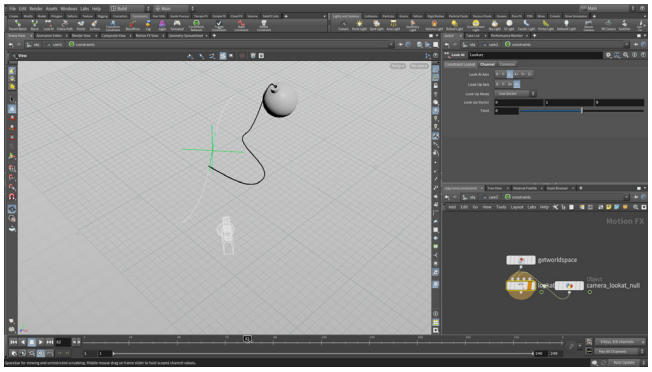
04 拘束ネットワークが Null オブジェクトに追加され、path ノードが作成されました。**フレーム 1** に移動します。Path ノードを選択したまま、**Position** パラメータを RMB クリックして **Delete Channels** を選択し、パスに沿って Null をアニメートしているエクスプレッションを削除します。**Position** を **0** に設定します。**Position** を **Alt クリック**して、キーフレームを設定します。

フレーム 195 に移動して、**Position** を **1** に設定します。**Position** を **Alt クリック**して、2 つ目のキーフレームを設定します。

Animation Editor タブをクリックし、**H を押して**カーブ全体を表示します。カーブを選択し、**Straight** ボタンを押して直線にします。2 つ目のポイントの接線を使用して、最後が滑らかになるように調整します。

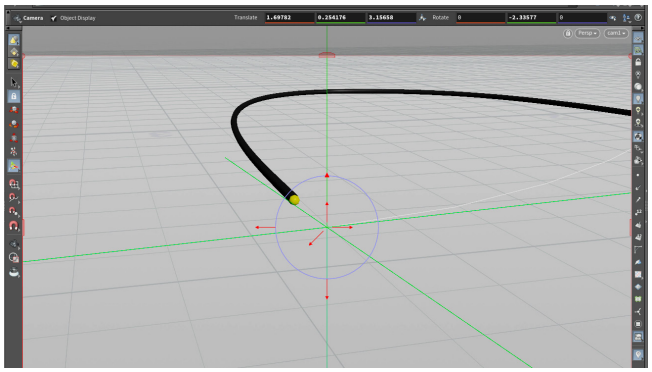


05 ネットワークビューで、**Tab > Camera** を選択し、**Enter** を押して、クリックして原点に配置します。**Move** ツールを使用して、カメラを導火線の前の少し右側に移動します。Y 軸に沿って上に移動し、地面から約 **0.75** ユニット上げます。



06 **Constraints** シェルフから、**Look At** ツールをクリックします。これにより、選択したカメラが **look at** オブジェクトとして使用されます。***null*** オブジェクトを **look at** オブジェクトとして選択し、**Enter** を押します。再度 **Enter** を押して、**look up** オブジェクトとしては何も割り当てないようにします。

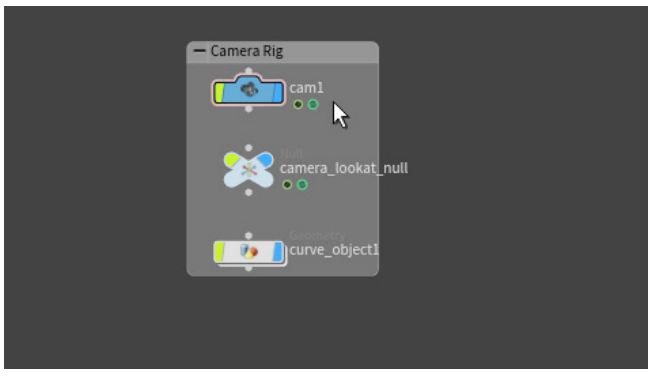
これで、カメラが Null オブジェクトに向くようになりました。Camera メニューから ***cam1*** を選択し、このカメラ越しに見てみましょう。



07 **フレーム 1** に移動します。camera を選択して、**Handle** ツールがアクティブなことを確認します。カメラハンドルが表示されるので、ハンドルを使用して導火線の開始点がよく見えるようにカメラを再配置します。

フレーム 195 に移動します。同じハンドルを使用して、爆弾がフレーム内の適切な位置に見えるようにカメラを再配置します。

シーケンスを通してカメラが適切に動作するように、前後にスクラブして微調整しましょう。シーケンス全体を通して、導火線が見えるようにします。



08 オブジェクトレベルに移動して、***curve***、***camera_lookat_null*** オブジェクトと ***cam1*** ノードを選択して整列させ、ネットワークボックスに入れます。ボックスのタイトルバーをダブルクリックし、ボックスの名前を ***Camera Rig*** にします。すべてのパーツの Display フラグを**オフにして**、作業時に Scene View に表示されないようにします。このボックスは必要に応じて折り畳んだり、展開したままにすることができます。

シーンを**保存**します。

拘束

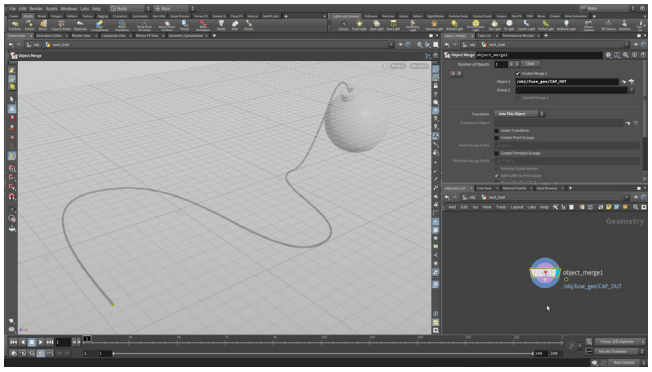
Null オブジェクトをパスに追従させ、カメラをパスに向かせるために、**Constraints** シェルフにあるアニメーション拘束を使用しました。これらは、**チャンネルオペレータ**または **CHOP** と呼ばれる特別なノードタイプを使って行われます。これらのノードは、null ノードや camera ノードの内部にあります。これらを使用して、拘束の動作を制御することができます。

CHOP ノードは、任意のパラメータを **RMB クリック**すると表示される **Motion FX** メニューでも使用できます。

パート 5

すすのトレイルの作成

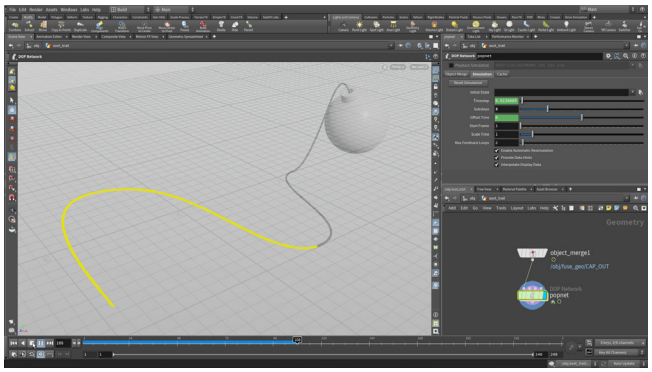
すすのトレイルを作成するには、キャップを使用してパーティクルトレイルを放出します。これらのポイントを適切に放出する方法と、重力などのフォースを追加してパーティクルの動きを制御する方法を学びます。また、衝突をセットアップして、パーティクルが地面にぶつかったり、爆弾のサーフェスを滑り落ちるようにする方法も学びます。



01

fuse_geo オブジェクトのノード中に入ります。***CAP_OUT*** の Display フラグをオンにした状態で、**Modify** シェルフに移動して、**Extract** ツールを使用します。**N を押して**すべてのフェースを選択し、**Enter** を押すと、Object Merge ノードを使ってキャップを取り込んだ新しいオブジェクトが作成されます。オブジェクトレベルに移動して、このオブジェクトの名前を ***soot_trail*** にします。

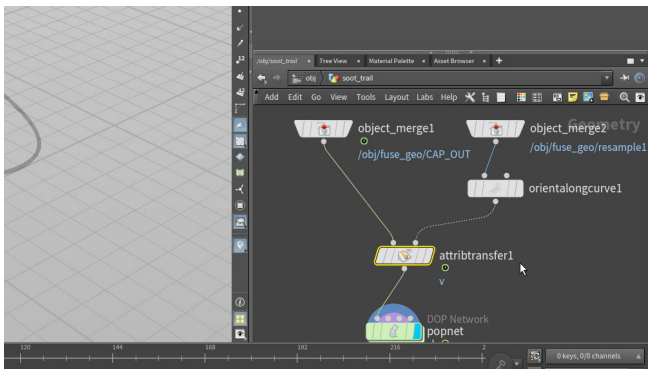
fuse_geo オブジェクトに戻り、FUSE_OUT に Display フラグを設定します。これで、レンダリング時には導火線のジオメトリ全体が表示され、パーティクルの生成には新しいオブジェクトが使用されます。



02

soot_trail ノードの中に入り、POP Network ノードをチェーンの終端に追加します。その中に入り、***Source First Input*** ノードで、**Const Birth Rate** を **1000** に設定します。**Play** を押します。パーティクルが放出されるのを確認できますが、ただそれだけです。

1 つ上のレベルに戻り、**Simulation** タブで **Substeps** を **3** に設定します。**Play** を押します。パーティクルがより均一に放出されているのが分かります。

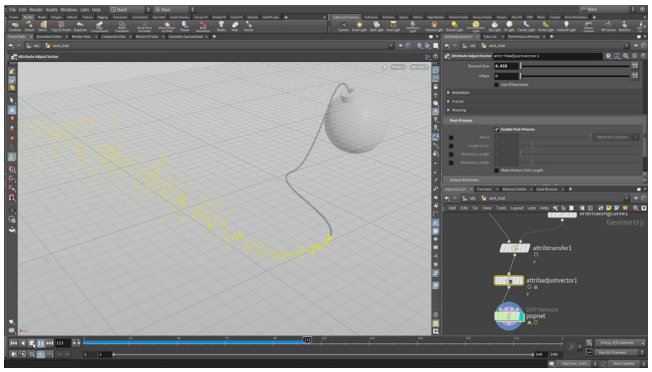


03

object_merge ノードを **Alt ドラッグ**して、2 つ目のコピーを作成します。**Object 1** を ***fuse_geo>resample*** ノードに設定します。このカーブを使用して、キャップに Velocity を転送します。

Orient Along Curve ノードを追加したら、**Output Attributes** の **Tangent (Z Axis)** をオンにして、**v** に設定します。

オリジナルの **object_merge** ノードと ***popnet*** ノードの間に ***Attribute Transfer*** ノードを追加します。**Primitives** チェックボックスをオフにして、**Points** を **v** に設定します。

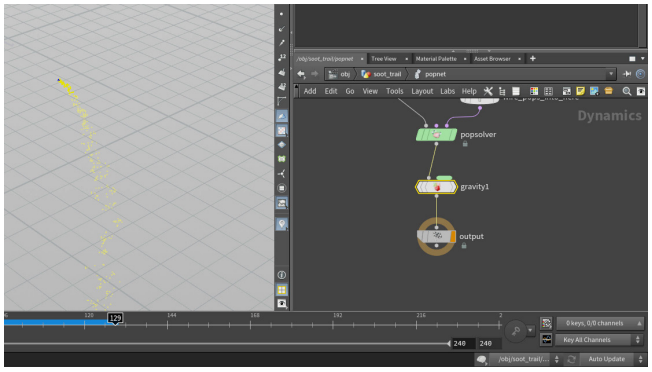


04

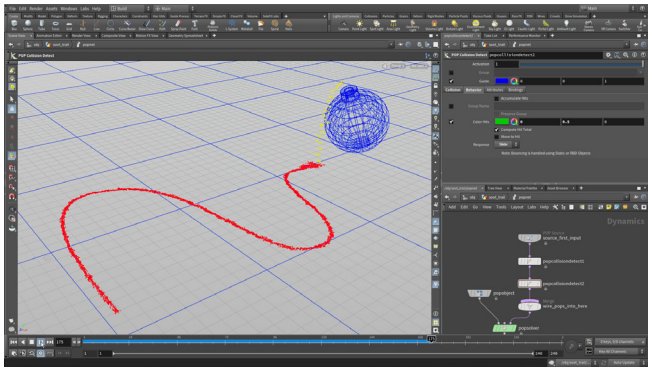
attributetransfer と popnet の間に **Attribute Adjust Vector** ノードを配置します。Adjustment Value で次のように設定します。

- **Adjustment for** を **Direction Only** にする
- **Adjust with** を **Noise** にする
- **Range Values** を **Zero Centered** にする
- **Amplitude** を **0.5** にする

Noise Pattern の **Element Size** を **0.025** に設定し、**Post-Process** の **Enable Post-Process** をオンにします。**Play** を押してテストします。



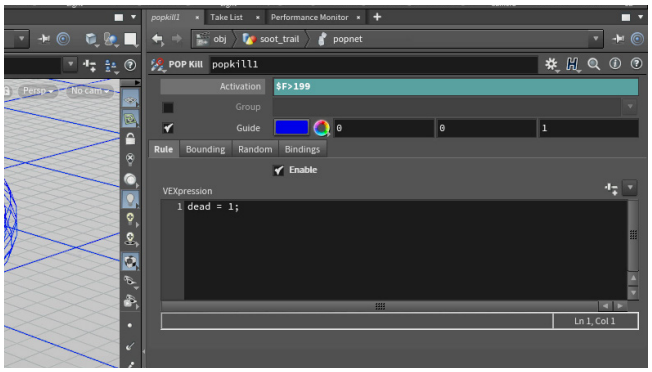
05 ***popnet*** の中に入って、***popsolver*** の下に **Gravity Force** ノードを追加します。これでシミュレーションを**再生**すると、パーティクルが地面に落ちるようになります。



06 オブジェクトレベルに戻り、**グリッド**を作成します。**Size** を **30, 30**、**Rows** と **Columns** を **31, 31** にします。

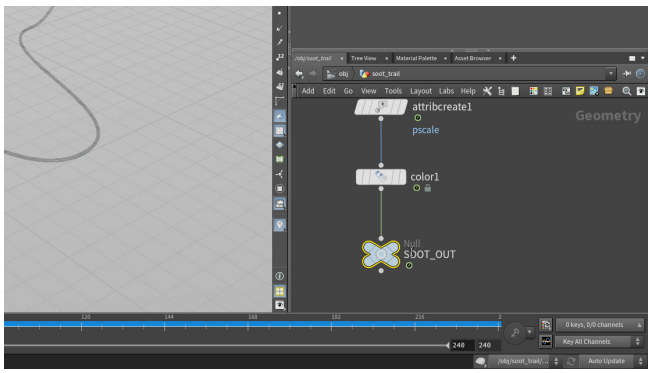
名前を ***ground*** に変更します。***popnet*** の中に入り、**Pop Collision Detect** ノードを ***source_first_input*** ノードの後に追加します。**SOP** Path を `obj/ground/grid1` に設定します。**Behavior** タブで、**Response** を **Stick** に設定します。**Color Hits** は**赤色**のままにしておきます。

Pop Collision Detect ノードをもう 1 つ追加します。SOP Path を ***bomb_geo*** ジオメトリオブジェクトに設定します。**Response** を **Slide** に設定します。**Color Hits** を**緑色**に変更します。**Play** を押します。



07 現在は、パーティクルがシーケンス全体を通して放出されています。爆弾が爆発する瞬間に、パーティクルを止めなければなりません。**POP Kill** ノードを ***wire_pops_into_here*** merge ノードの後に追加します。フレーム 1 に移動して、**Activation** を **$F>199** に設定します。これで、パーティクルはフレーム 200 で終了します。

Rule タブで Enable をオンにします。**Play** を押してテストします。



08 ジオメトリコンテキストに戻り、**attribute create** ノードを作成して popnet に接続します。**Name** を **pscale** に、**Value** を **0.001** に設定します。

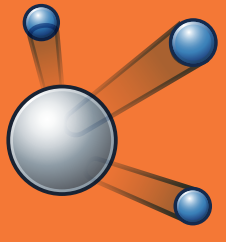
Color SOP を追加して、**Color** を**ダークグレー**に設定します。

null ノードをチェーンの終端に追加して、名前を ***SOOT_OUT*** に変更します。

パーティクル FX

すすのトレイルは、パーティクルダイナミクスを使用して作成します。パーティクルとは、風や重力といったフォースを使用して影響を与えることのできるポイントです。導火線の末端から、ポイントを発生させ、さまざまなテクニックでシミュレートすることができます。

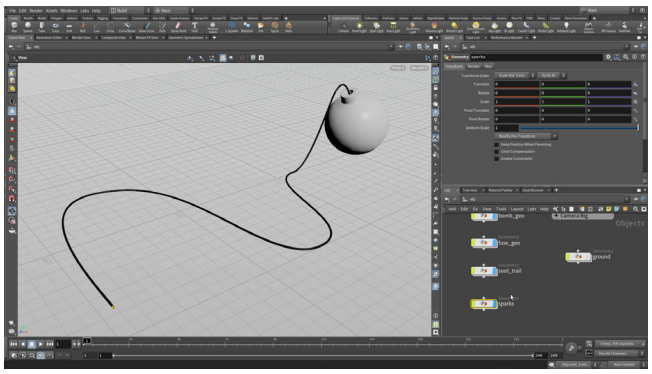
パーティクルは Houdini の Dynamics または DOP セクションを使用してシミュレートした後、SOP に戻してジオメトリとして扱うことができます。次のセクションでは、パーティクルを使用して導火線の末端に火花を作成します。



パート 6

パーティクルの火花の作成

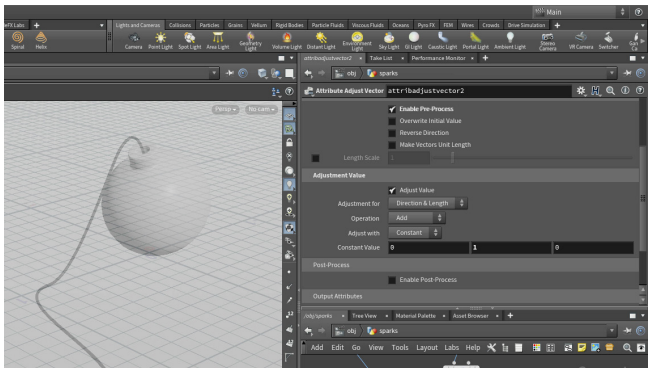
火花を作成するには、すすのパーティクルオブジェクトをコピーして、火花を生成する新しいオブジェクトとなるように設定を変更します。火花のパーティクルの方が寿命が短く、活発になるようにします。Spark Trail ノードを使用すると、希望通りのルックの火花をショットに追加することができます。このノードのパラメータを調整することで、目的のルックを実現します。



01

フレーム 1 に戻り、**オブジェクト**レベルに移動します。**Ground** オブジェクトを非表示にします。***soot_trail*** オブジェクトを **Alt ドラッグ** して、コピーを作成します。コピーの名前を ***sparks*** にします。

この新しいオブジェクトには既に popnet があり、これを火花のパーティクルシミュレーションを生成するように変更します。Houdini では、すべてをゼロから作成するのではなく、既にあるネットワークを再利用することをお勧めします。



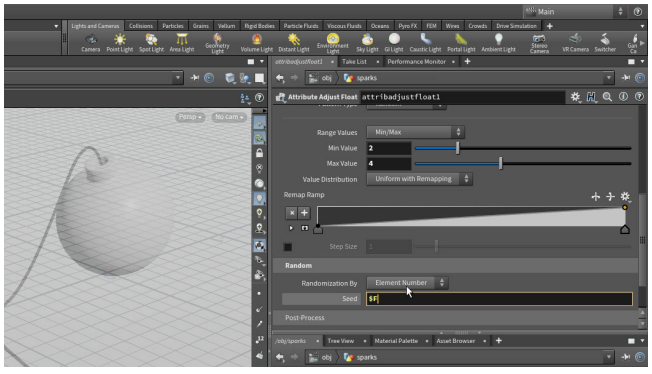
02

sparks オブジェクトに入ります。いくつか変更を加えて、火花を作成するネットワークをセットアップします。

atttributecreate と ***color*** ノードを**削除**します。これらはこのパーティクルネットワークには必要ありません。null ノードの名前を ***SPARKS_OUT*** に変更します。

attributeadjustvector ノードで、**Amplitude** を **1.75** に変更します。

このノードの上に、新しい **Attribute Adjust Vector** ノードを追加します。**Enable Pre-Process** をオンにして、**Constant Value** を **0, 1, 0** に設定します。これで、パーティクルが上昇してから落下するようになりました。**Play** を押します。



03

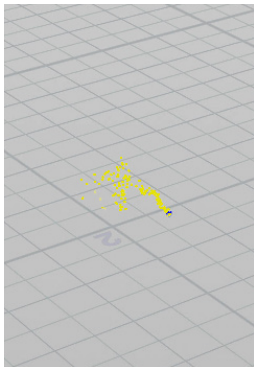
フレーム 1 に戻ります。**Attribute Adjust Float** ノードを ***popnet*** ノードの直前に追加します。次のように設定します。

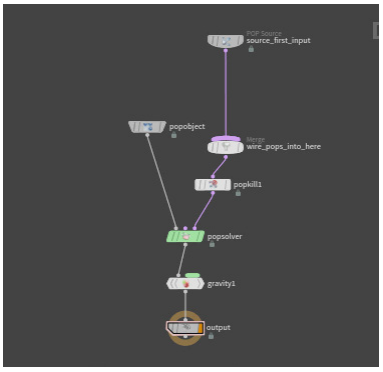
- **Attribute Name** を **life** にする
- **Unit Settings** を **Duration** にする
- **Pattern Type** を **Random** にする
- **Min Value** を **2**、**Max Value** を **4** にする
- **Random** で、**Seed** を **$F** にする

火花 | PARTICLE TRAIL

Particle Trail SOP は、アニメーションするパーティクルシステムを受け取り、そのパーティクルからモーショントレイルを生成します。これらのトレイルは、火花、花火、雨などの様々なエフェクトで使用可能です。

このノードは、そのトレイルのルックを制御することもできます。これによって、長い期間のモーションブラーを使ってポイントをレンダリングすることなく、SOP コンテキスト内でルックを微調整することができます。

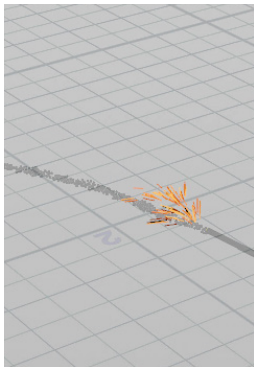


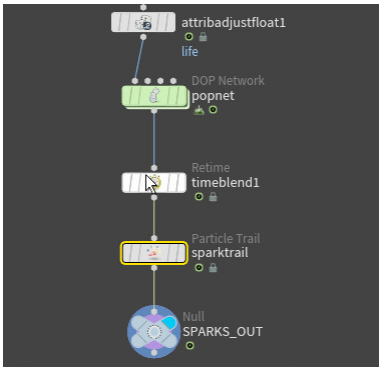


04

popnet の中に入り、2 つの ***collisiondetect*** ノードを **削除**します。この時点では衝突を気にする必要はありません。後から必要に応じて別のノードを追加します。

Play を押して、パーティクルが放出されるのを確認します。



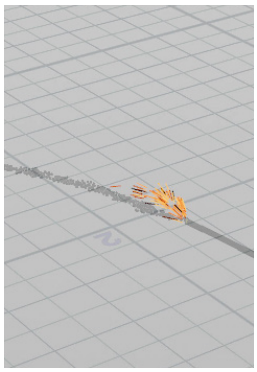


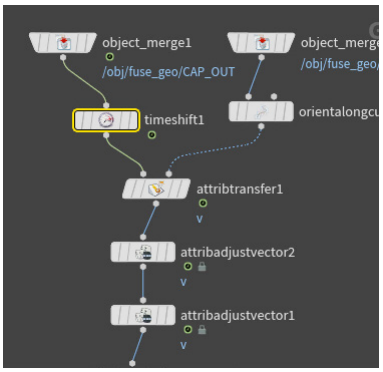
05

ジオメトリレベルに戻ります。フレーム 1 に移動します。

ネットワークビューで、**Tab > Spark Trail** を押します。少し遅れて、***timeblend*** と ***sparktrail*** の 2 つのノードが追加されます。***popnet*** を ***timeblend*** ノードに接続し、***sparktrail*** を ***SPARKS_OUT*** Null に接続します。

Play を押します。火花が見えますが、導火線の末端よりも遅れています。

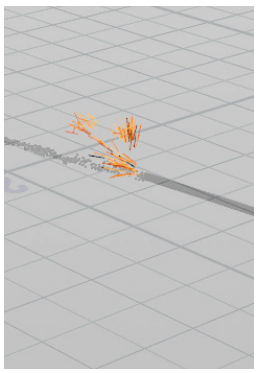


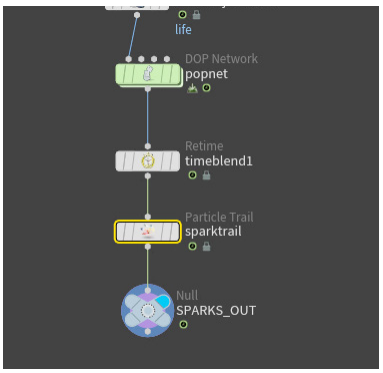


06

キャップを取り込む ***object_merge*** と、***attributetransfer*** ノードの間に Time Shift ノードを追加します。**Frame** パラメータ名をクリックして、使用されているエクスプレッションを確認します。それを `$FF + 1` に変更します。

これによりキャップが 1 フレーム分進むため、火花と導火線の末端が揃うようになります。**Play** を押してテストします。



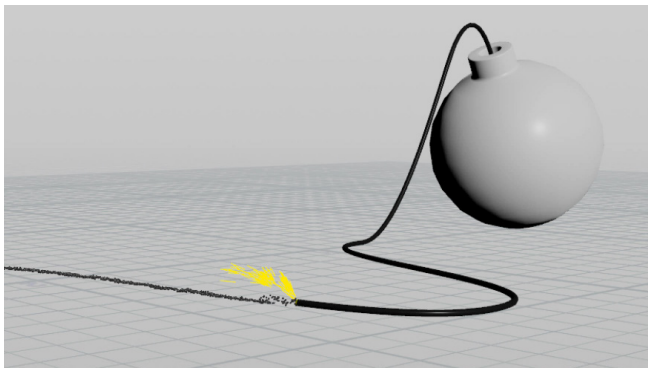


07

フレーム 1 に移動し、***sparktrail*** ノードを選択します。**Split** タブをクリックし、**Enable Split** チェックボックスをオンにします。**Percent to Split** を **40** に設定します。

Shape の **Splits per Point** を **4** に設定します。

Play を押してテストします。



08

フレーム 1 に戻り、オブジェクトレベルに移動します。

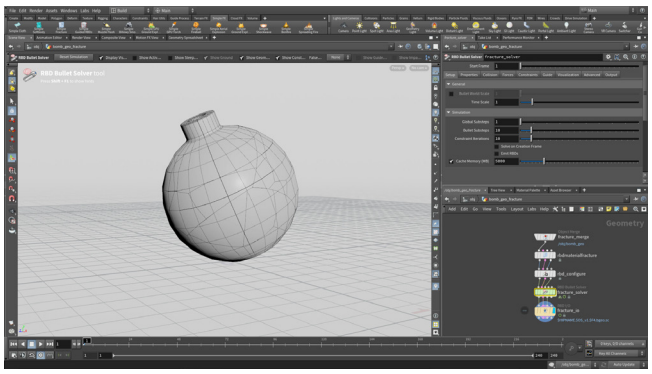
カメラを ***cam1*** に設定し、ツールバー下部の **Flipbook** ボタンを押します。表示されるウィンドウで **Start** をクリックします。Scene View のアニメーションシーケンスが作成され、パーティクルの動きを評価できます。

Realtime トグル がオンになっていることを確認したら、フリップブックを再生して、ショットの進行を確認します。

パート 7

爆弾の爆発

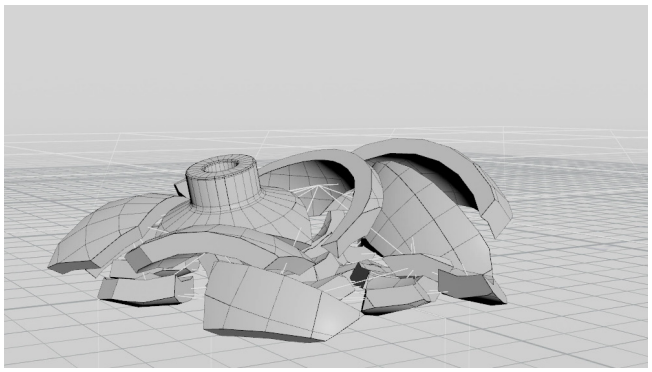
爆弾のジオメトリに必要なのは、リジッドボディダイナミクスシミュレーションです。ジオメトリを粉砕してから、アトリビュートを追加して爆発を作成します。その後、移動するパーツの速度を制御して、芸術的な観点でルックを調整します。シミュレーションの準備が整ったら、ジオメトリをキャッシュして、効率的に PyroFX の段階に進めるようにします。



01

fuse_geo、***soot_trail***、***sparks*** オブジェクトを非表示にします。**スペースバー + G** を押して、爆弾のジオメトリを中心に寄せます。プレイバーで**フレーム 1** に移動します。**スペースバー + G** を押して、爆弾を中心に寄せます。

bomb_geo オブジェクトを選択し、**Simple FX** シェルフから **Simple Fracture** を選択します。***衝突オブジェクト***の選択を求められたら、何も選択せずに **Enter** を押します。すると、このネットワークがすぐにセットアップされます。このオブジェクトは爆弾のジオメトリを結合して、粉砕とシミュレーション用のノードをセットアップします。

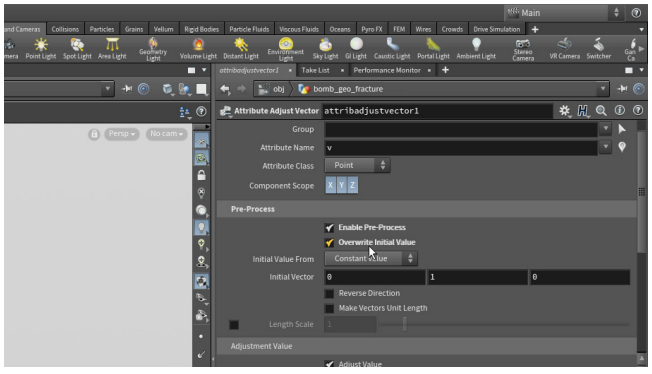


02

fracture_solver ノードで、**Start Frame** を **200** に設定します。**プレイバー**の開始フレームを **200** に設定します。**First Frame** ボタンをクリックし、フレーム 200 に移動します。爆発のシミュレーションが必要なのはフレーム 200 から 240 のみです。

fracture_solver ノードで **Collision** タブに移動して、**Ground Collision** セクションの **Ground Type** を **Ground Plane** に設定します。**Advanced** タブに移動し、**Constraints** > **Glue** セクションで、**Data Name** フィールドの ***Glue*** という語を削除します。

Play を押して、爆弾がバラバラになるのを確認します。次は、パーツを爆発させるために開始 Velocity を追加します。

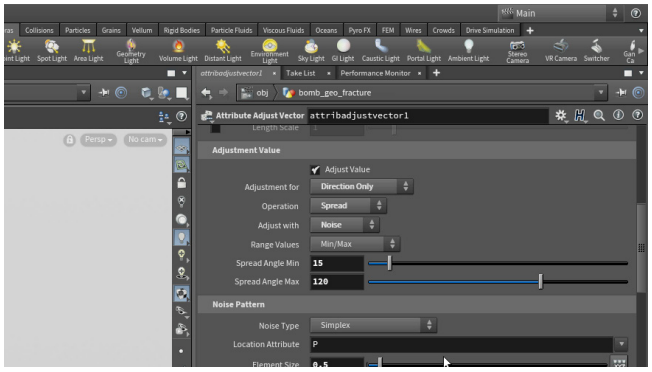


03

フレーム 200 に移動します。**Attribute Adjust Vector** ノードを追加して、他のノードの右側に配置します。***rbd_configure*** ノードの 3 つ目の出力を ***attributeadjustvector*** ノードに接続し、***attributeadjustvector*** ノードを ***fracture_solver*** ノードの 3 つ目の入力に接続します。

このノードで次のように設定します。

- **Enable Pre-Process** を**オン**にする
- **Overwrite Initial Value** を**オン**にする
- **Initial Vector** を **0, 1, 0** にする



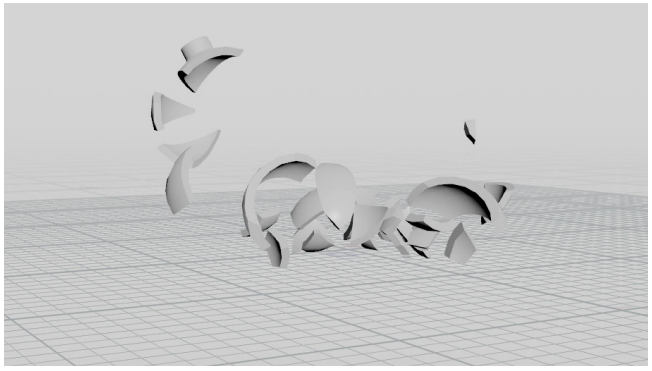
04

Adjustment Value で、次のように設定します。

- **Adjustment for** を **Direction Only** にする
- **Operation** を **Spread** にする
- **Adjust with** を **Noise** にする
- **Spread Angle Min** を **15** にする
- **Spread Angle Max** を **120** にする

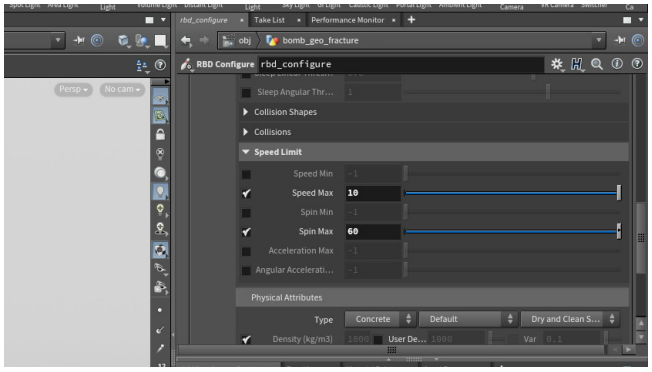
Noise Pattern で **Element Size** を **0.5** に設定します。

Post-Process で **Enable Post-Process** をオンにしたら、**Minimum Length** をオンにして、**20** に設定します。



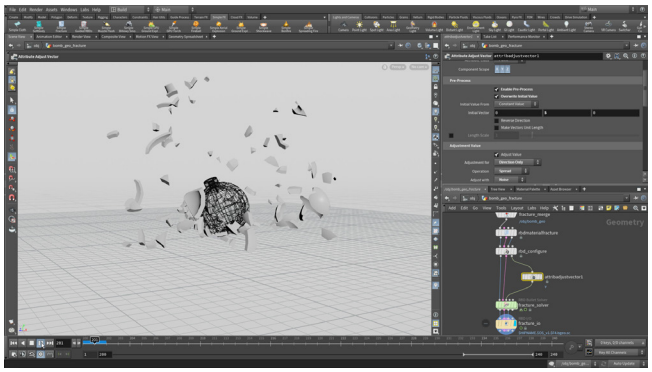
05 **Play** を押します。爆弾が爆発するようになりました。ジオメトリに対する Velocity アトリビュートの設定がシミュレーションに送られ、破片の初期 Velocity を使用して破片を押し動かします。

Houdini では、多くのビジュアルエフェクトのセットアップで、アトリビュートの操作が効果的です。



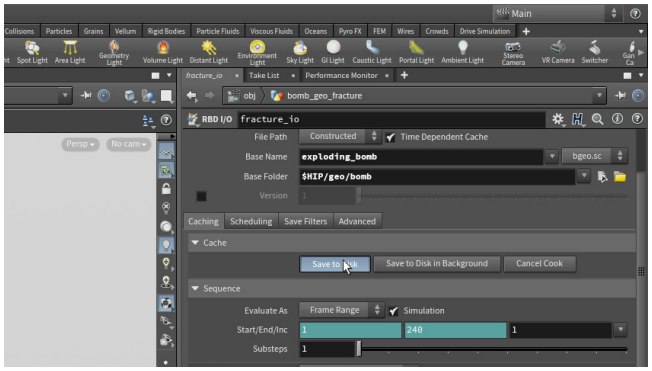
06 ***rbdconfigure*** ノードを選択します。**Speed Limit** セクションを開き、**Speed Max** と **Spin Max** パラメータをオンにします。**Speed Max** を **2**、**Spin Max** を **30** に設定します。**Play** を押します。シミュレーションが元の速度よりもだいぶ遅くなります。

今度は、**Speed Max** を **10**、**Spin Max** を **60** に設定します。**Play** を押します。

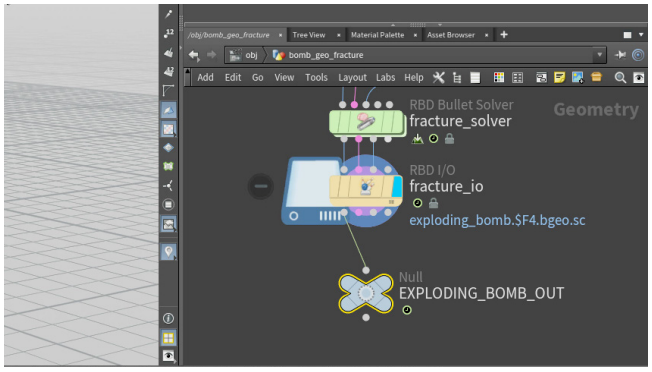


07 rbdmaterialfracture ノードを選択して、**Cell Points** の **Scatter Points** を 25 に設定します。**Play** を押します。これで、破片がかなり増えました。

attributeadjustvector ノードに戻り、**Length Scale** を **50**、**Initial Vector** を **0, 5, 0** に設定します。**Play** を押します。



08 **フレーム 200** に移動します。fracture_io ノードで、**Base Name** を ***exploding_bomb***、**Base Folder** を ***$HIP/geo/bomb*** に設定します。**Save to Disk** をクリックすると、**Load From Disk** がオンになります。**Play** を押して、キャッシュ化されたジオメトリをプレビューします。

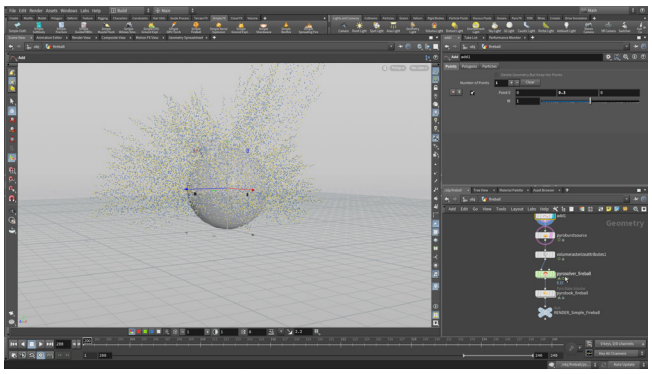


09 **Null** ノードを追加して、それに ***fracture_io*** ノードの 1 つ目の出力を接続します。このノードを ***EXPLODING_BOMB_OUT*** という**名前に変更**します。

パート 8

PyroFX の爆発の作成

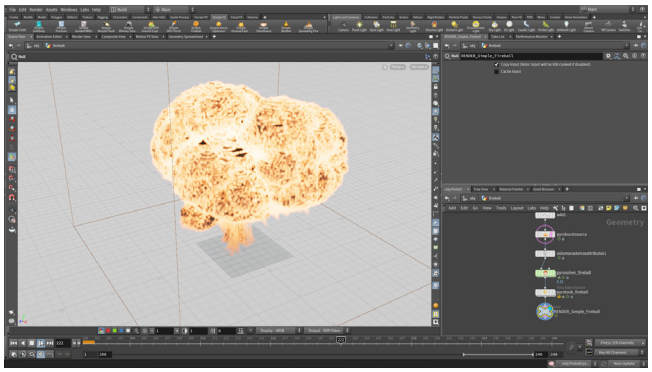
爆弾の爆発には、火の玉が伴います。GPU で動作する Simple Fireball から始め、変更を加えてショットにふさわしいルックを作成します。また、爆発する爆弾のパーツを組み込んで、PyroFX ボリュームを押したり、作用したり、さらに面白い見た目にすることもできます。



01 **オブジェクト**レベルに移動します。**Simple FX** シェルフから、**Simple Fireball** をクリックします。Scene View で、**Enter** を押して原点に配置します。これにより、***fireball*** オブジェクトの内部に複数のノードが作成されます。

pyroburstsource ノードに **Display フラグ**を設定し、**Burst Animation** タブで **Start Frame** を **200** に設定します。**フレーム 200** に移動します。このノードは、爆発の最初の爆風です。

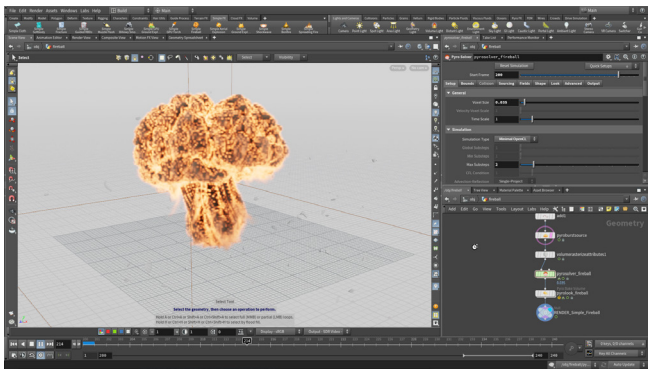
Quick Setups メニューから、**Single Input Point** を選択します。これにより、1 個のポイントが追加されます。このトランスフォームハンドルを使用して、ポイントを約 0.3 引き上げ、爆弾のジオメトリの中央に配置します。



02 ***pyrosolver_fireball*** ノードで、**Start Frame** を **200** に変更します。**Simulation Type** を **Minimal OpenCL** に設定し、GPU を使用してシミュレートするようにします。

Sourcing タブで、**Limit Source Range** オプションを開き、**Frame Range** を **200, 240** に設定します。**Cycle Length** をオフにします。

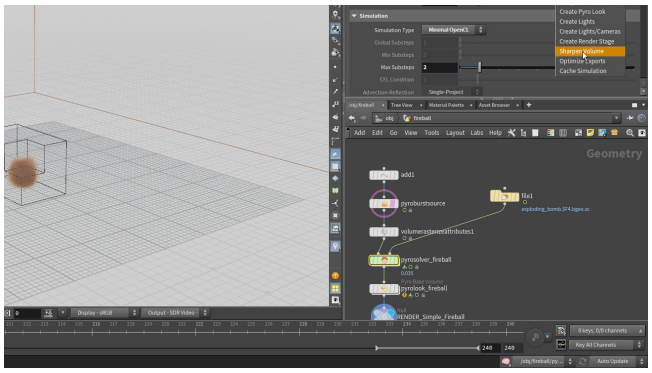
RENDER_Simple_Fireball Null ノードに **Display フラグ**を設定します。ズームアウトしてシーンを広く見えるようにしたら、**Play** を押して、シミュレーションをテストします。ずいぶん大きい火の玉です。さらにズームアウトして爆発全体を表示します。



03 **フレーム 200** に移動します。***pyroburstsource*** ノードで、**Burst Shape** タブに移動して **Initial Size** を **0.35**、**Spread Angle** を **180** に設定します。

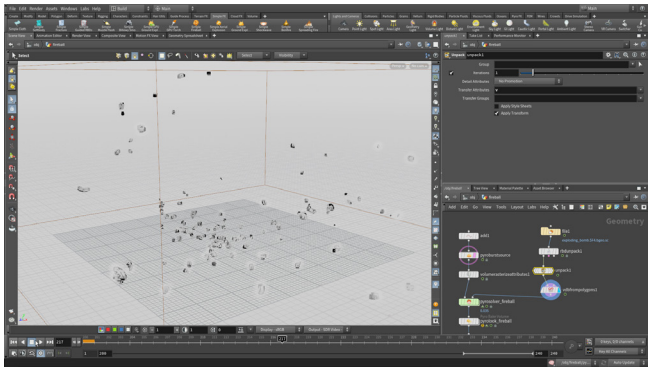
次に ***fireball*** の Pyro Solver ノードに移動して **Bound** タブで **Size** を **15, 12, 12**、**Center** を **0, 4, 0** に設定します。ボックスが小さくなり、GPU 上で効率的にシミュレートできるようになります。**Setup** タブで、**Voxel Size** を **0.035** に設定します。これで、小さくしたシミュレーションにディテールが追加されます。

Play を押して、シミュレーションをテストします。爆発がシーンにうまく収まるようになりました。



04 次に、Pyro FX と爆発する爆弾を統合します。**File** ノードを追加して、ディスクから ***$HIP/geo/bomb/exploding_bomb.$F.bgeo.sc*** ジオメトリシーケンスをロードします。このノードを ***fireball*** の Pyro Solver ノードの 2 つ目の入力に接続します。

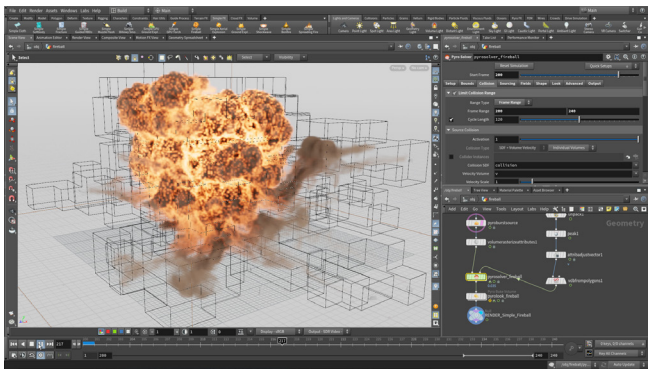
Play を押して、シミュレーションをテストします。適切に準備されていないので、衝突ジオメトリにエフェクトがありません。



05 フレーム **200** に移動します。***fireball*** ノードを選択します。右上の **Quick Setups** メニューから、**Setup SDF Collision** を選択します。これにより、***vdbfrompolygons*** ノードが追加されます。このノードの **Voxel Size** は Pyro Solver ノードから取得されます。

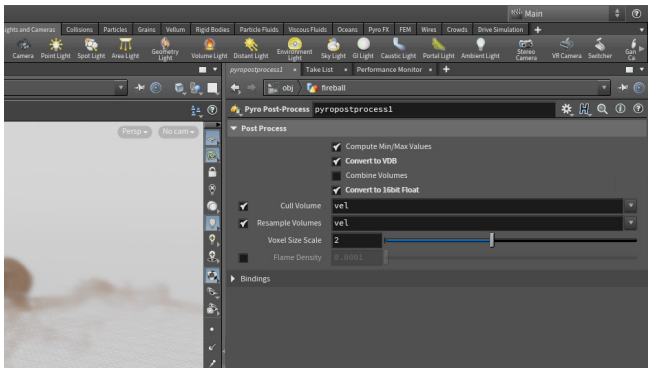
file ノードと ***vdbfrompolygons*** ノードの間に **RBD Unpack** ノードを追加します。

rbdunpack ノードと ***vdbfrompolygons*** ノードの間に **Unpack** ノードを追加します。***unpack*** ノードで、**Transfer Velocity** を **v** に設定します。***vdbfrompolygons*** ノードに **Display フラグ**を設定し、**Play** を押して衝突ジオメトリを表示します。



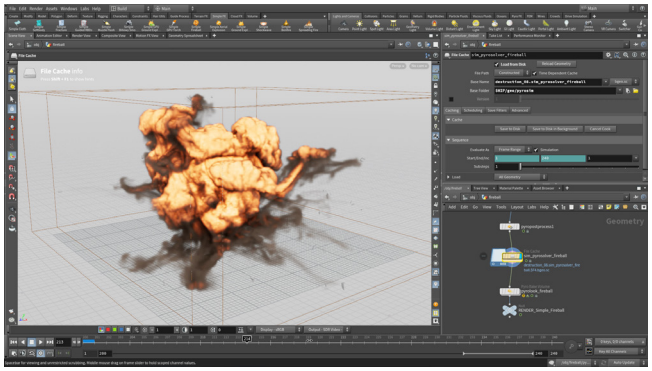
06 ***unpack*** と ***vdbfrompolygons*** の間に、**Peak** ノードを追加して破片を大きくします。**Distance** を **0.1** に設定します。***peak*** ノードと ***vdbfrompolygons*** ノードの間に、**Attribute Adjust Vector** ノードを追加します。**Adjust Value** をオフ、**Enable Post Process** をオンにしてから、**Length Scale** をオンにして **2** に設定します。

fireball ノードに **Display フラグ**を再度設定します。**Collision** タブから、**Limit Collision Range** セクションを開き、**Range Type** を **Frame Range**、**Frame Range** を **200, 240** に設定します。**Cycle Length** をオフにします。**Play** を押して、シミュレーションをテストします。爆発する爆弾にコリジョンが構築され、Pyro FX シミュレーションに影響するようになりました。



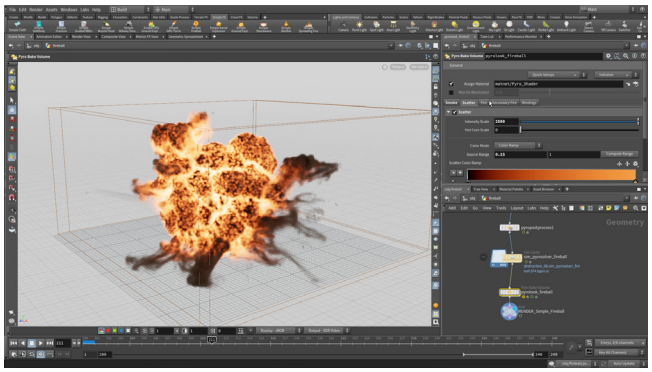
07 ***fireball*** の Pyro Solver ノードと ***pyrolook*** ノードの間に、**Pyro Post Process** ノードを挿入します。**Convert to VDB** と **Convert to 16bit Float** チェックボックスをオンにします。次に、**Cull Volume** と **Resample Volumes** オプションをオンにして、どちらも ***vel*** のままにしておきます。

このノードにより、ボリュームがより効率的になり、ボリュームをキャッシュ化する際のディスク容量を節約できます。



08 ***fireball*** の Pyro Solver ノードで、**Quick Setups** をクリックし、**Cache Simulation** を選択します。そのノードを横に移動して、***pyropostprocess*** ノードを sim_pyrosolver_fireball File Cache ノードに接続し、cache ノードを ***pyrolook*** ノードに接続します。Base Folder を ***$HIP/geo/pyrosim/*** に設定し、**Save to Disk** をクリックします。

Load from Disk オプションをオンにしたまま、プレイバーをスクラブして PyroFX の爆発を確認します。



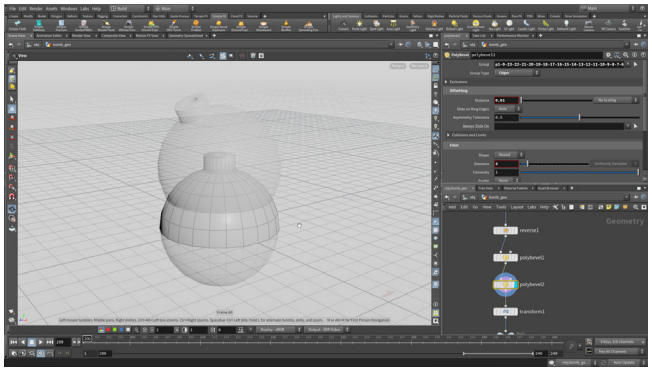
09 ***pyrolook*** ノードに Display フラグを設定します。これは **Pyro Bake Volume** ノードで、Scene View でシミュレーションを視覚化するのに使用できます。後でレンダリング時に使用する、Pyro Shader とインターフェースがよく似ています。

Smoke タブで **Smoke Color** を暗くして、**Scatter** タブで **Intensity Scale** を **2500** に設定します。他の調整も行って、シミュレーションが目的のルックになるようにします。

パート 9

ジオメトリを USD にエクスポート

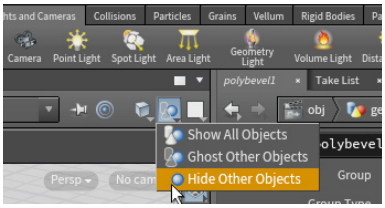
レンダリング向けにショットをセットアップするには、USD ファイルにジオメトリをエクスポートして、Solaris コンテキストで参照できるようにする必要があります。ジオメトリを直接インポートすることもできますが、USD としてキャッシュ化すると、シーケンスを固定して、Solaris でライティングとレンダリングに集中できます。これらのオブジェクトの一部では、エクスポートする前に UV を追加して、テクスチャリングに備えます。

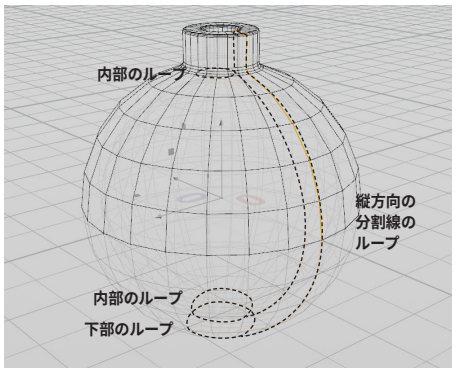


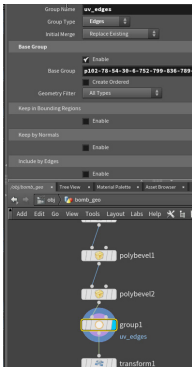
01

オブジェクトレベルに移動して、***fireball*** オブジェクトを非表示にします。***bomb_geo*** をダブルクリックし、2 つ目の ***polybevel*** ノードに **Display フラグ**を設定します。これにより、所定の位置に回転される前の状態の爆弾のジオメトリが表示されます。

Hide Other Objects を設定し、このネットワークのコンテンツに集中できるようにします。





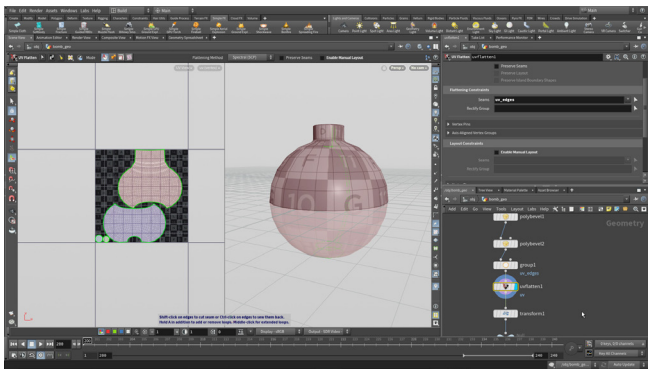


02

Select ツールに移動し、**3 を押してエッジ**選択にします。**W を押して**ワイヤーフレームモードにしたら、**Shift** を押しながら**ダブルクリック**して下部のループの部分を選択します。**Shift** を押しながら**ダブルクリック**して、4 つのすべての領域を選択します。内部ループでも繰り返します。

内部の球の上部にあるループに移動し、**Shift** を押しながら**ダブルクリック**して選択します。X 軸と平行な横方向の分割線を、**Shift** を押しながら**ダブルクリック**して選択します。

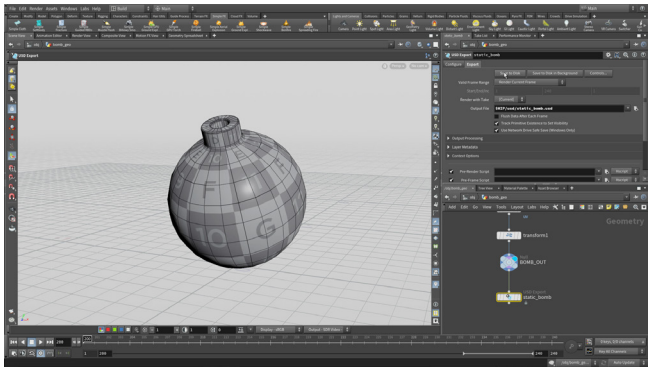
Scene View で、**Tab > Group** を押します。**Group Name** を ***uv_edges*** に変更します。



03

Scene View で、**Tab > UV Flatten** を押すと、group ノードの後に ***uvflatten*** ノードが追加されます。**Seams** を ***uv_edges*** に設定し、**Layout Constraints** で **Enable Manual Layout** をオフにします。こうすると UV レイアウトがよく見えるため、爆弾のテクスチャリングに好都合です。

BOMB_OUT null ノードに **Display フラグ**を設定します。

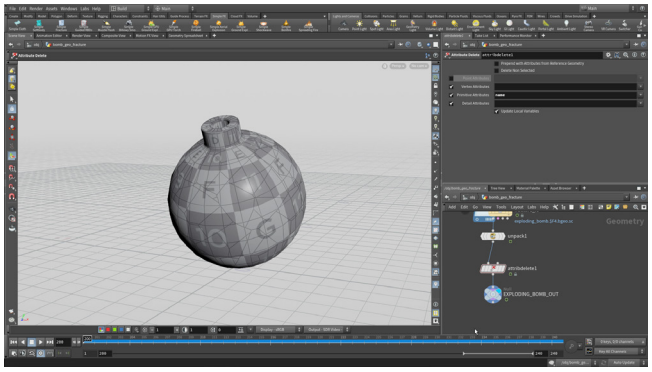


04

チェーンの終端に **USD Export** ノードを追加します。***static_bomb*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Current Frame** にする
- **Output File** を ***$HIP/usd/static_bomb.usd*** にする

Save to Disk を押します。

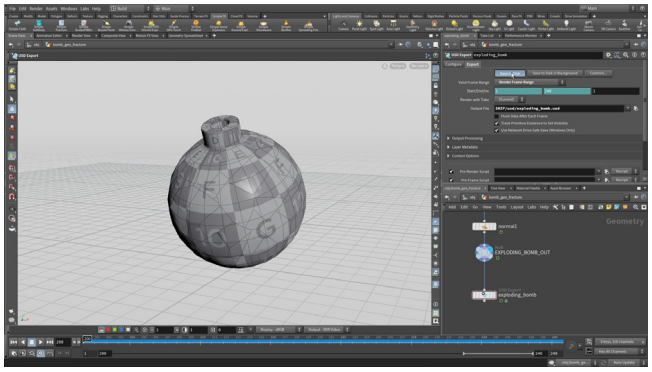


05

オブジェクトレベルに戻り、***bomb_geo_fracture*** ネットワークをダブルクリックします。***fracture_io*** ノードを選択し、**Save to Disk** をクリックして、新しい UV 付きでジオメトリをキャッシュ化します。ジオメトリはパック化されているため表示されません。

fracture_io ノードと ***EXPLODING_BOMB_OUT*** ノードの間に **Unpack** ノードを追加したら、**Attribute Delete** ノードを追加します。***attributedelete*** ノードで、**Primitive Attributes** を ***name*** に設定します。**Point Attributes** セクションをオフにします。

これにより、シーケンスが単一のメッシュとして Solaris に取り込まれます。name アトリビュートはシーケンスを個々のパーツに分割します。



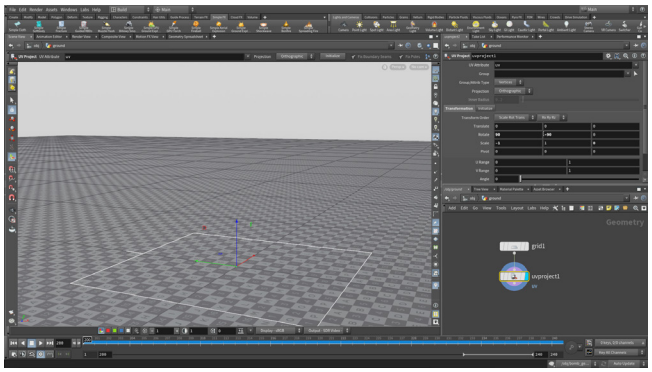
06

attributedelete ノードの後に、**Normal** ノードを追加します。これにより、爆弾のジオメトリが Solaris で適切に表示されるようになります。

チェーンの終端に **USD Export** ノードを追加します。***exploding_bomb*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/exploding_bomb.usd*** にする

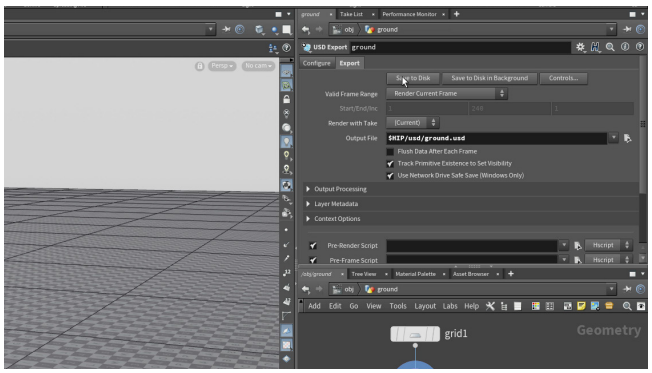
Save to Disk を押します。



07

オブジェクトレベルに戻り、***ground*** ネットワークを**ダブルクリック**します。***grid*** ノードの後に、**UV Project** ノードを追加して、**Display フラグ**を設定します。**Initialize** タブに移動して、**Initialize** ボタンを押します。**Transformation** タブで、**Scale X** を **-1**、**Scale Y** を **1**、**Rotate Y** を **-90** に変更します。

これにより、1 つの大きいテクスチャが作成されるのではなく、テクスチャが地面で繰り返されるようになります。



08

チェーンの終端に **USD Export** ノードを追加します。それを ***ground*** という**名前に変更**します。次のように設定します。

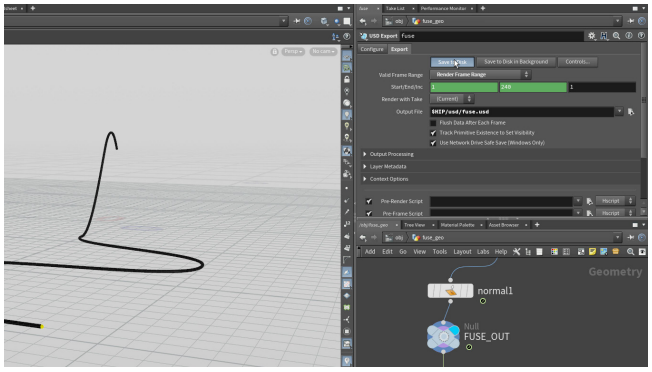
- **Valid Frame Range** は **Render Current Frame** のままにする
- **Output File** を ***$HIP/usd/ground.usd*** にする

Save to Disk を押します。

USD と SOLARIS

このプロジェクトのルックデベロップメントをサポートするために、レイアウト、ルックデブ、ライティングワークフローは **Solaris** コンテキストでセットアップします。これは、**LOP ネットワーク**で表現されます。ここで作成する **USD** キャッシュを、Solaris コンテキストに取り込みます。

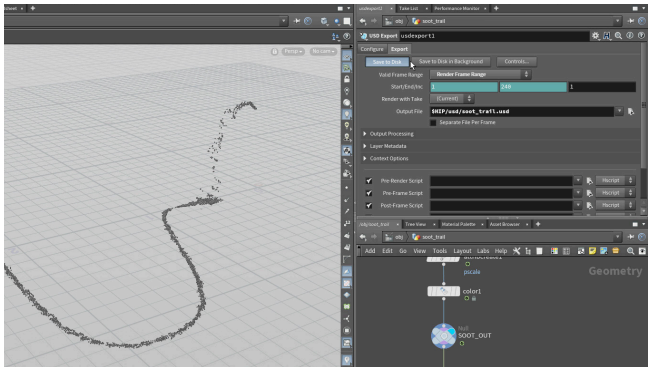
このシーンファイルを使用して USD キャッシュを LOP ネットワークで参照しますが、より大規模なパイプラインでは、新しいシーンファイルを開始し、新しいシーンで USD ファイルをインポートする方法もあります。こうするとショットのライティングとレンダリングに集中できますが、ジオメトリとシミュレーションに戻って微調整するのは難しくなります。



09 オブジェクトレベルに戻り、***fuse*** ネットワークを**ダブルクリック**します。チェーンの終端に **USD Export** ノードを追加します。それを ***fuse*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/fuse.usd*** にする

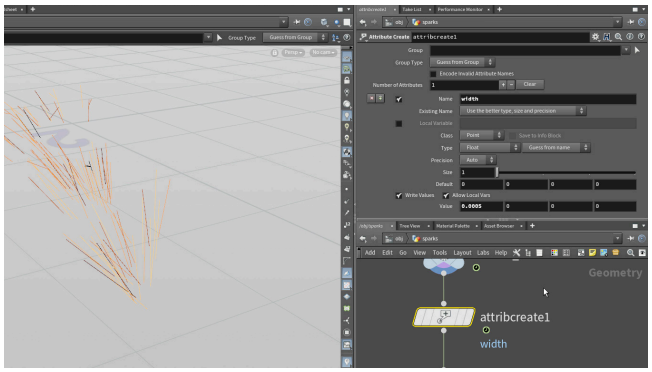
Save to Disk を押します。



10 オブジェクトレベルに戻り、***soot*** ネットワークを**ダブルクリック**します。チェーンの終端に **USD Export** ノードを追加します。それを ***soot_trail*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/soot_trail.usd*** にする

Save to Disk を押します。



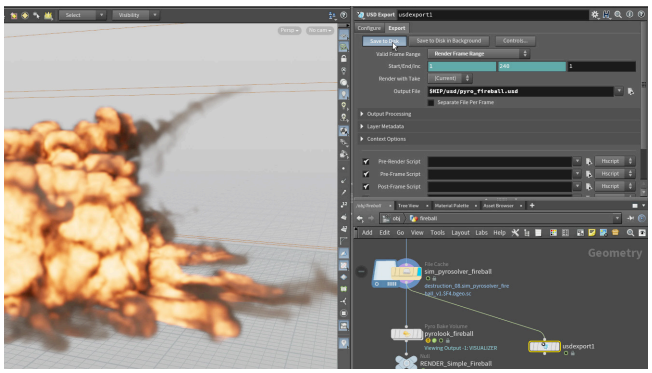
11 オブジェクトレベルに戻り、***sparks*** ネットワークを**ダブルクリック**します。

チェーンの終端に **Attribute Create** ノードを追加します。**Name** を ***width***、**Value** を **0.0005** に設定します。これにより、レンダリングされる火花のルックが決まります。

チェーンの終端に **USD Export** ノードを追加します。それを ***sparks*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/sparks.usd*** にする

Save to Disk を押します。



12 オブジェクトレベルに戻り、***fireball*** ネットワークを**ダブルクリック**します。***sim_fireball*** ノードから、**USD Export** ノードを分岐させます。***pyrolook*** ノードをバイパスしてください。それを ***pyro_fireball*** という**名前に変更**します。次のように設定します。

- **Valid Frame Range** を **Render Frame Range** にする
- **Output File** を ***$HIP/usd/pyro_fireball.usd*** にする

Save to Disk を押します。

Scene Import

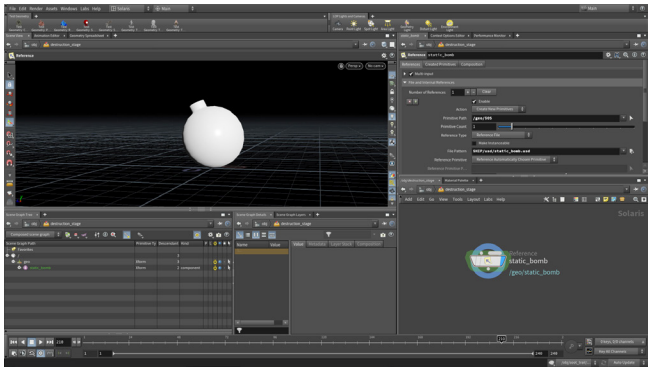
ジオメトリとシミュレーションを Solaris に取り込むもう 1 つの方法は、**Scene Import LOP** を使用することです。これにより、作業しているジオメトリおよびオブジェクトと、LOP ネットワークの間に、直接的なつながりができます。この方法では、モーションブラーをサポートする Cache LOP が必要になります。この追加のノードは、USD ファイルを参照する場合には必要ありません。

このツールは、アニメーションカメラをオブジェクトレベルから Solaris コンテキストに取り込むのに使用します。

パート 10

Solaris でのショットのセットアップ

すべての USD ファイルを Solaris で参照して、オブジェクトレベルからカメラをインポートする方法を学びます。マテリアルをすべての要素に適用し、Karma を使用してレンダリングを開始して、結果を評価します。また、キーライトとレンダリング設定を準備して、ショットの最終ルックを探求します。

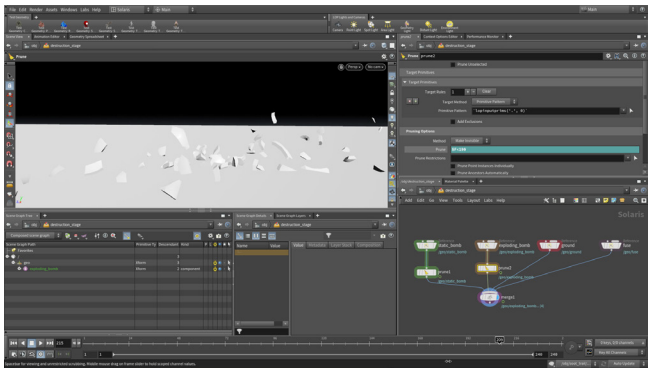


01

オブジェクトレベルに戻ります。**Tab > LOP Network** を押して、ショットをセットアップするのに使用するサブネットワークを作成します。名前を ***destruction_stage*** にします。そのノードを**ダブルクリック**して、中に入ります。

デスクトップを **Solaris** に変更します。Scene View のパスバーに ***obj > destruction_stage*** が表示されていることを確認します。Scene View で **D を押して**、**Background** タブで **Color Scheme** を **Dark** に設定します。

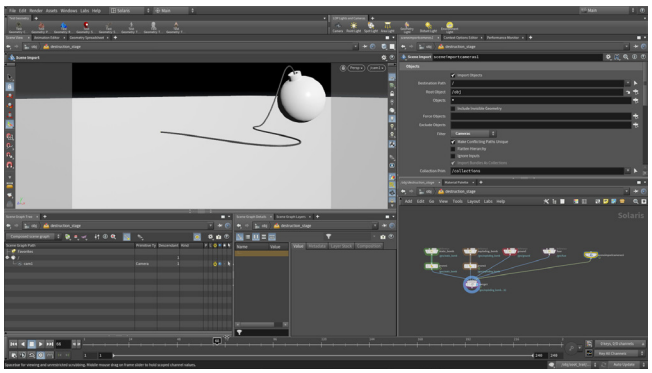
ネットワークビューで **Tab > Reference** を選択し、クリックして ***reference*** ノードを追加します。**Reference File** の横にある **File Chooser** をクリックし、***static_bomb.usd*** ファイルを指定します。ノードを ***static_bomb*** という**名前に変更**します。**Primitive Path** を ***/geo/$OS*** に設定します。



02

このノードを **Alt ドラッグ**してコピーを 3 つ作成します。それらに ***exploding_bomb***、***ground***、***fuse*** という名前を付け、**File Chooser** パラメータをこれらの USD ファイルに設定します。

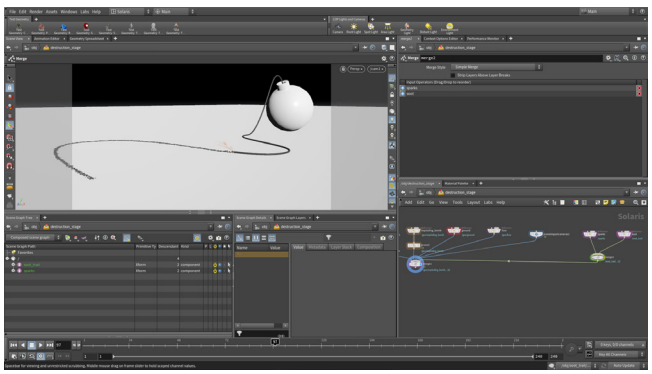
Merge ノードを追加してこれらを接続し、**Display フラグ**を設定します。***static_bomb*** ノードの後に、**Prune LOP** を追加します。**Prune** パラメータを ***$F >200*** に設定します。***expoding_bomb*** ノードの後に、**Prune LOP** をもう 1 つ追加します。**Prune** パラメータを ***$F <199*** に設定します。これで、プレイバーをスクラブすると、フレーム 200 で静的な爆弾と爆発する爆弾が切り替わります。



03

Tab > Scene Import (Cameras) を押します。このノードを配置して、***merge*** ノードに接続します。**Destination Path** を ***/cam/*** に設定します。これにより、オブジェクトレベルの任意のカメラを Solaris コンテキストに追加できます。

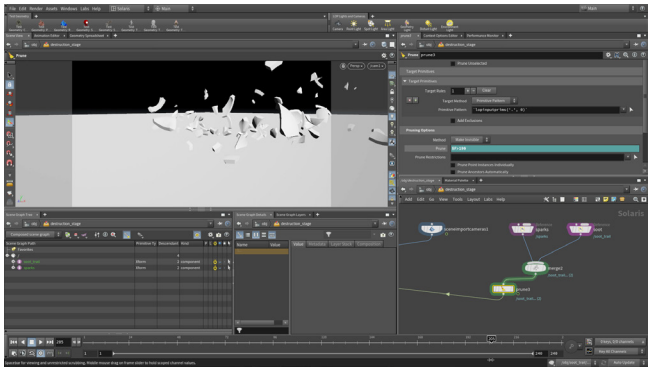
カメラメニューに移動して、***cam1*** を選択し、このアニメーションカメラ越しに見るようにします。merge ノードが選択されていることを確認します。プレイバーをスクラブし、オブジェクトレベルでセットアップしたカメラのレンズを通して、導火線と爆弾がアニメートされ、爆弾が爆発することを確認します。このカメラに変更を加えた場合は、ここ Solaris で反映されます。



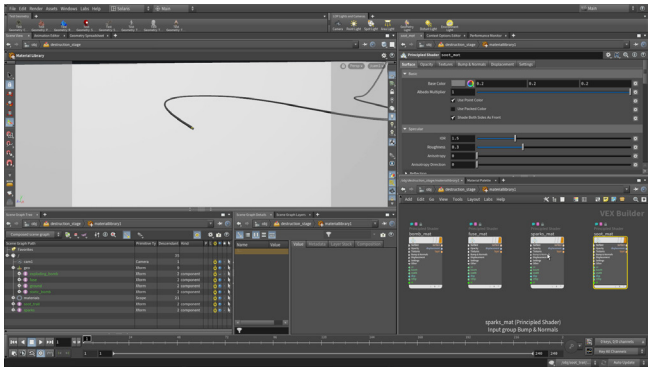
04

ネットワークビューで **Tab > Reference** を選択し、クリックして ***reference*** ノードを追加します。**Reference File** の横にある **File Chooser** をクリックし、***sparks.usd*** ファイルを指定します。ノードを ***sparks*** という名前に変更します。**Primitive Path** を ***/fx/$OS*** に設定します。

このノードを **Alt ドラッグ**してコピーを作成します。**soot** という名前を付けたら、**File Chooser** をクリックして ***soot_trail.usd*** ファイルを指定します。**Merge** ノードを追加してこれらのファイルをまとめたら、それをメインの **Merge** ノードに接続し、**Display フラグ**を設定します。スクラブすると、火花が見え、火花が消えた後にすすが残るのが分かります。USD ファイルはフレーム 240 までパーティクルを維持します。

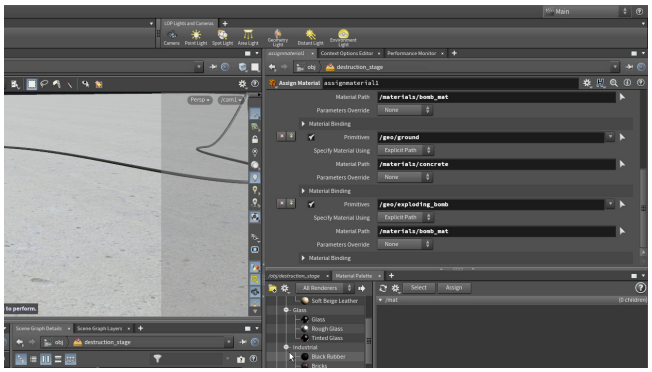


05 エフェクト ***merge*** ノードの後に、**Prune LOP** を追加します。**Display フラグ**を設定します。**Prune** パラメータを **$F >199** に設定します。これで、この時点ですると火花が消えるようになります。



06 ***merge*** ノードの後に **Material Library** ノードを追加します。**Material Palette** に移動して、**Principled Shader** と **Concrete Shader** を ***/stage/materiallibrary*** にドラッグします。

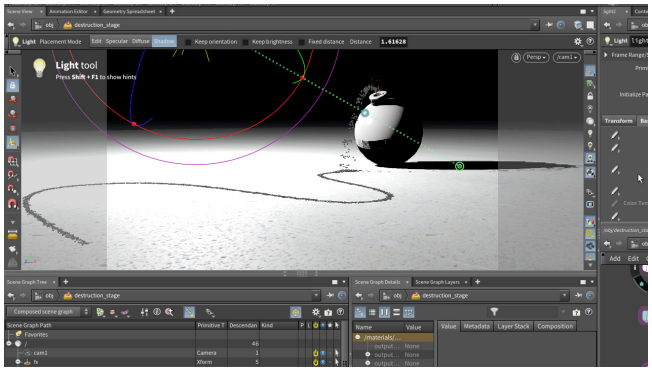
ネットワークビューに移動して、Principled Shader の名前を ***bomb_mat*** に変更します。**Alt ドラッグ** して Principled Shader のコピーを 3 つ作成し、***fuse_mat***、***sparks_mat***、***soot_mat*** という名前にします。



07 1 つ上のレベルに戻り、チェーンの終端に **Assign Material** ノードを追加します。シーングラフから、***static_bomb*** と ***exploding_bomb*** を **Primitives** フィールドにドラッグします。

Material Path の横の矢印をクリックして、***bomb_mat*** を選択します。**+（プラス）記号**を押してセクションを追加し、この手順を繰り返して ***fuse***、***sparks***、***soot*** にマテリアルを割り当てます。

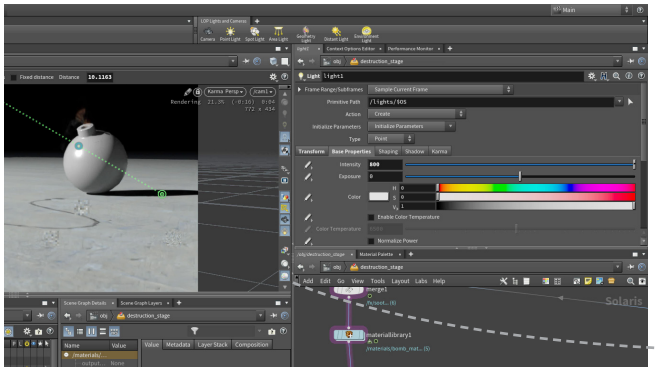
concrete マテリアルを ***ground*** に割り当てます。



08 フレーム 180 あたりに移動します。**LOP Lights and Cameras** シェルフの **Light** ツールをクリックします。ショットにライトが追加され、ライト越しに見えるようになります。

Base Properties タブに移動し、**Intensity** を **50** に設定します。

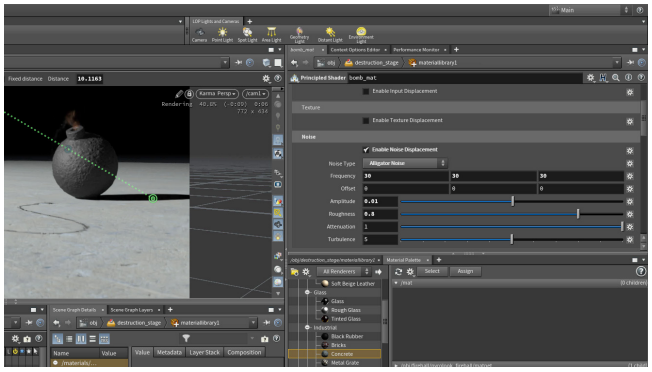
Scene View で、**Shadow** ボタンをクリックします。爆弾のサーフェスで **クリック**したら、その背後を **Shift クリック**してシャドウを作成します。



09 Scene view で、**Persp** メニューから **Karma** を選択します。**Reference Plane** をオフにします。

Material Palette に戻り、***concrete*** シェーダを選択します。**Texture** セクションで、**Effect Scale** を **0.005** に設定します。

サイドバーで **Denoiser** をオンにします。グラフィックスカード（nVidia カードのみ）を使用して、レンダリング時に、ノイズが素早く解決されるようになります。



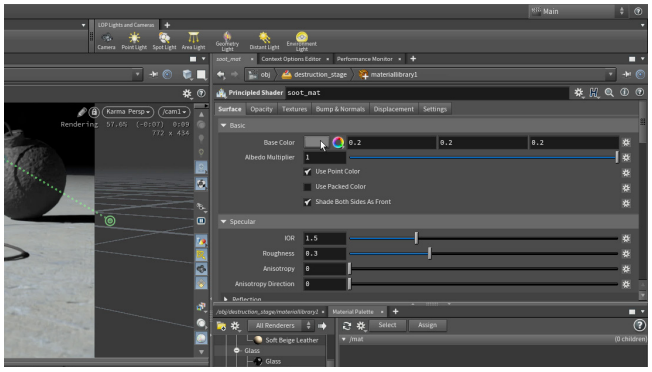
10

bomb_mat シェーダを選択します。**Surface** で次のように設定します。

- **Base Color** を**黒色 (0, 0, 0)** にする
- **Roughness** を **0.7** にする

Displacement で、**Enable Noise Displacement** をオンにし、次のように設定します。

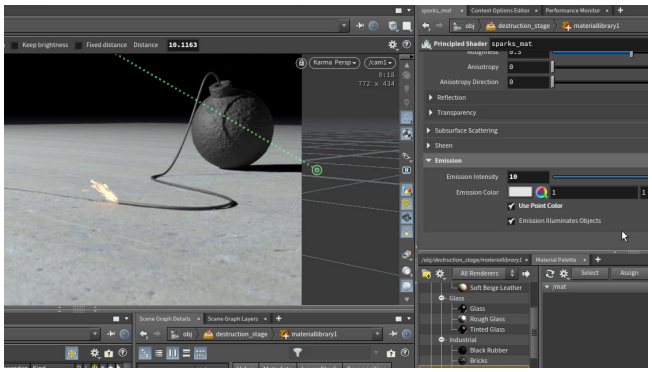
- **Noise Type** を **Alligator Noise** にする
- **Frequency** を **30, 30, 30** にする
- **Amplitude** を **0.01** にする
- **Roughness** を **0.8** にする



11

fuse_mat と ***soot_mat*** シェーダで、**Base Color** を***ダークグレー***に設定します。

fuse_mat シェーダで、**Use Point Color** をオフにして、Base Color でルックを制御できるようにします。

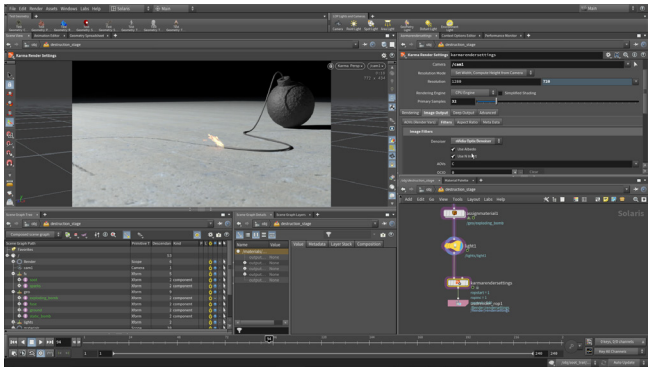


12

sparks_mat シェーダを選択し、**Emission** で次のように設定します。

- **Emission Color** を **1, 1, 1 (白)** にする
- **Emission Intensity** を **10** にする
- **Use Point Color** を**オン**にする

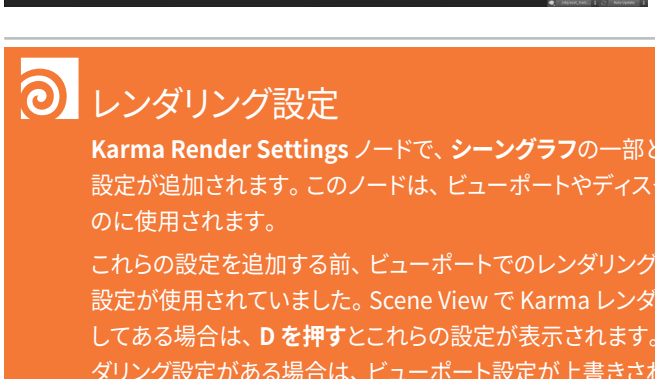
これで火花がより明るく光り、地面も少し照明されるようになります。



13

ステージレベルに戻ります。**ネットワークビュー**で、**Tab > Karma** を押し、**Karma Render Settings** と **USD Render ROP** ノードを追加します。それらをチェーンの終端に接続します。***karmarendersettings*** ノードを選択し、**Primary Samples** を **32** に設定します。**Image Output > Filters** タブで、**Denoiser** を **nVidia Optix Denoiser** に設定してデノイザをオンに戻します。

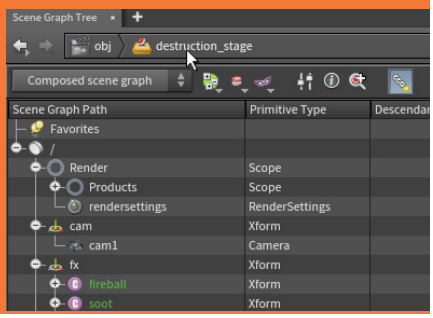
Advanced タブで、**Sampling** セクションに移動し、**Convergence Mode** を **Path Traced** に設定します。



レンダリング設定

Karma Render Settings ノードで、**シーングラフ**の一部となるレンダリング設定が追加されます。このノードは、ビューポートやディスクにレンダリングするのに使用されます。

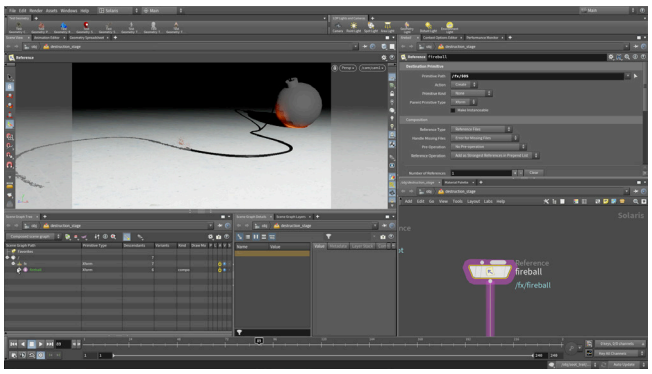
これらの設定を追加する前、ビューポートでのレンダリングにはビューポート設定が使用されていました。Scene View で Karma レンダリングをセットアップしてある場合は、**D を押す**とこれらの設定が表示されます。シーングラフにレンダリング設定がある場合は、ビューポート設定が上書きされます。



パート 11

PyroFX のレンダリング

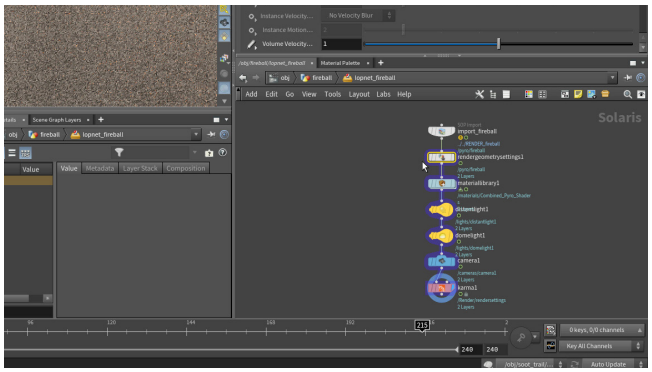
ショットを完成させるには、火の玉の USD ファイルを追加して、適切なマテリアルを割り当てる必要があります。次に、もう 1 つカメラをセットアップして爆発の広角ショットを作成したら、2 つのシーケンスをレンダリングして最終的なシーケンスに仕上げます。その後、Mplay 画像ビューアを使用して結果をプレビューします。



01

Persp ビューメニューを **Houdini GL** に再度設定します。ネットワークビューで **Tab > Reference** を選択し、クリックして ***Reference*** ノードを追加します。**Reference File** の横にある **File Chooser** をクリックし、***pyrofx_fireball.usd*** ファイルを指定します。ノードを ***fireball*** という名前に変更します。**Primitive Path** を ***/fx/$OS*** に設定します。

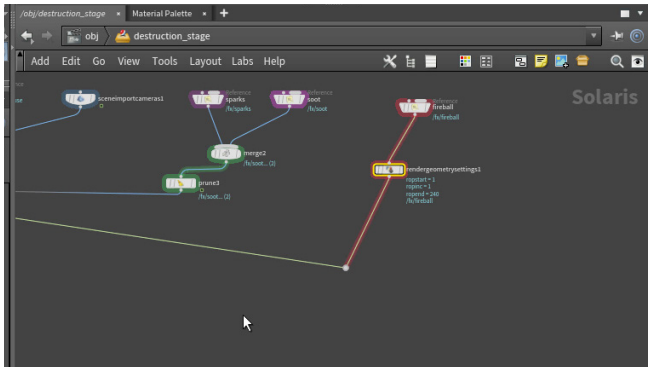
このノードを元の ***merge*** ノードに接続します。接続されたワイヤーを **Alt クリック**してドットを追加し、そのドットを右下に移動します。***Karma*** ノードの **Display フラグ**をオンにしたまま、**フレーム 204** あたりにスクラブして爆発を確認します。



02

オブジェクトレベルに移動して、***fireball*** オブジェクトに戻ります。***pyrosolver_fireball*** ノードを選択します。**Quick Setups** メニューから、**Create Render Stage** を選択します。これにより、このネットワークに ***lopnet_fireball*** が追加されます。その中に入り、推奨されるセットアップを確認します。これを単独で使用して火の玉をレンダリングすることもできますが、既存の LOP ネットワークの一部としてこのセットアップを使用する必要があります。

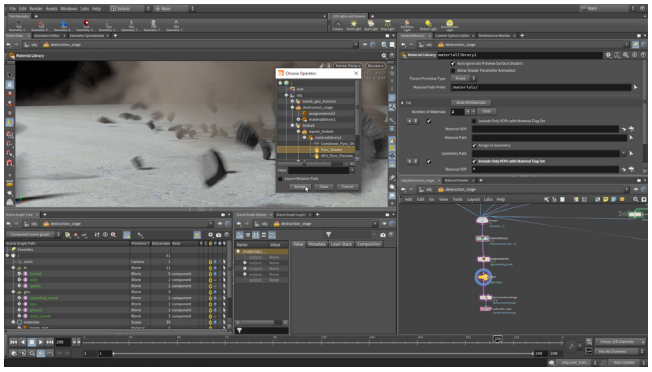
rendergeometrysettings ノードを選択し、**Ctrl + C** を押してコピーします。



03

destruction_stage LOP ネットワークに戻り、**Ctrl + V** を押してネットワークにペーストします。それを ***fireball*** reference ノードのすぐ下に接続します。

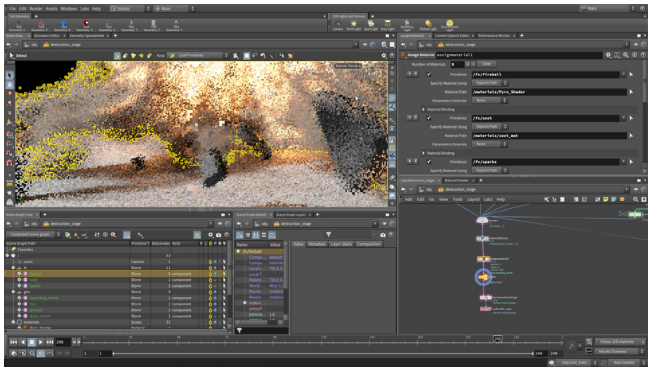
このノードは 2 つのことを行います。1 つ目は火の玉で Velocity モーションブラーをセットアップすることで、2 つ目はボリュームを使用してショットの照明をサポートすることです。



04

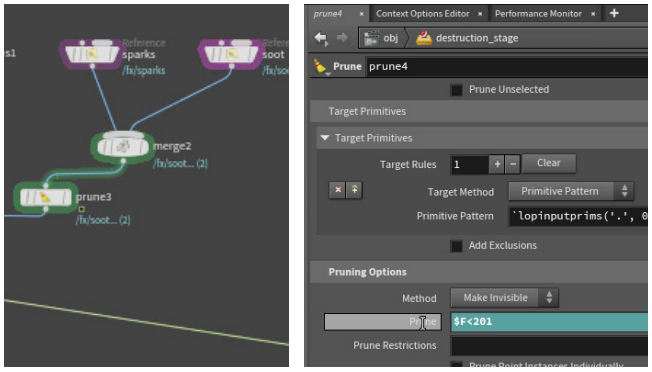
materiallibrary ノードを選択し、**Number of Materials** の横にある +(プラス)記号をクリックします。新しいリストの **Material VOP** の横にある Operator Chooser ボタンをクリックして、***fireball*** オブジェクト、***lopnet_fireball***、***matnet*** と移動していき、**Pyro_Shader** を選択します。**Accept** をクリックします。

このマテリアルは異なる LOP ネットワーク内にありますが、その位置からこの ***materiallibrary*** ノードに参照することができます。



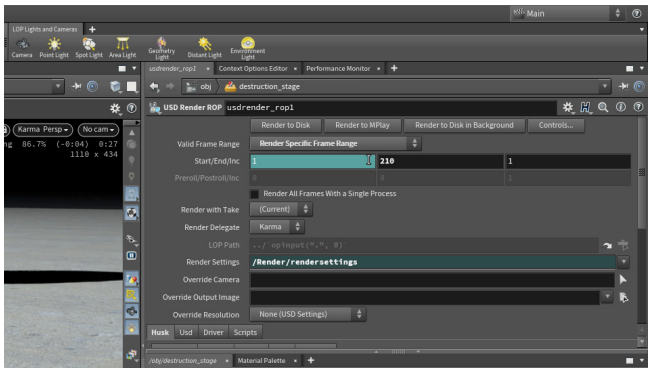
05 ***assignmaterial*** ノードで、マテリアルリストをもう 1 つ追加します。シーングラフから ***/fx/fireball*** を **Primitives** セクションにドラッグして、**Material Path** の矢印をクリックし、**Pyro_Shader** を選択します。

Persp ビューメニューを **Karma** に再度設定します。



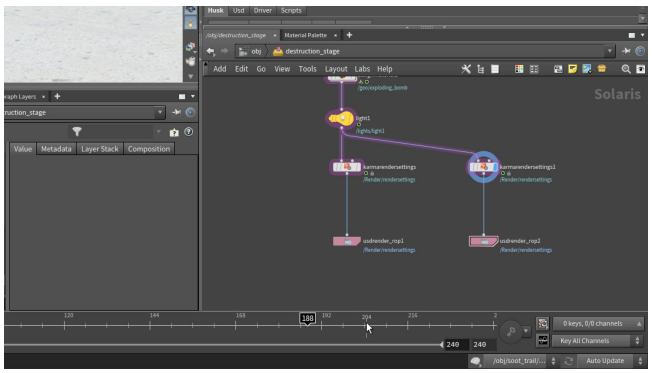
06 ネットワークに **Prune** ノードを追加し、***rendergeometrysettings*** ノードの下に接続します。Pruning Options で、**Prune** を **$F<201** に設定します。

フレーム 200 で火の玉が爆弾のジオメトリから出ていましたが、これにより爆発が 1 フレーム後ろにずれます。

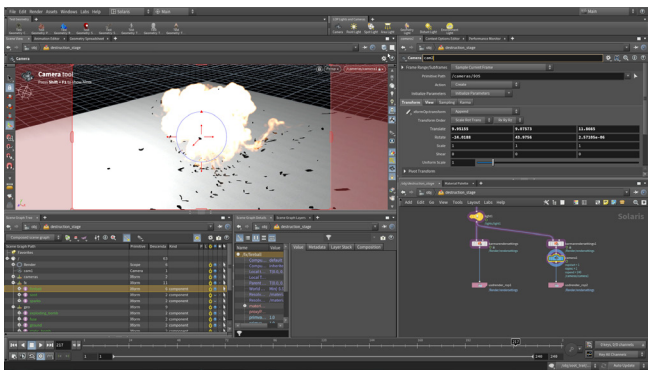


07 Karma ノードで、**Valid Frame Range** を **Render Frame Range** に設定します。**240** に設定されている **End** 値を **RMB クリック**し、**Delete Channels** を選択します。**End** 値を **210** に変更します。

最初の 210 フレームはアニメーションカメラを使用し、最後の 30 フレームは別のカメラに切り替えます。

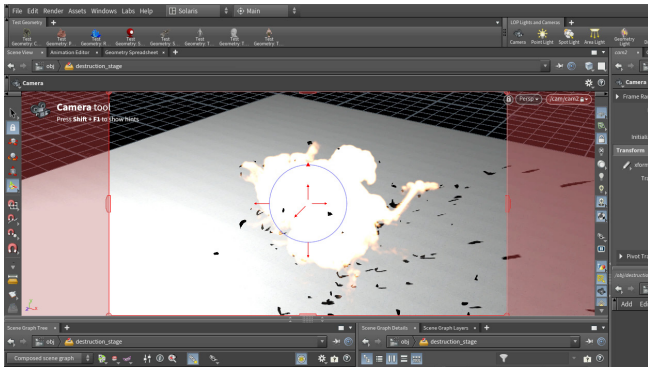


08 Persp ビューメニューを **Houdini GL** に戻します。ネットワークビューで、***karmarendersettings*** と ***usdrender_rop*** ノードを右側に **Alt ドラッグ**します。**Display フラグ**を設定します。

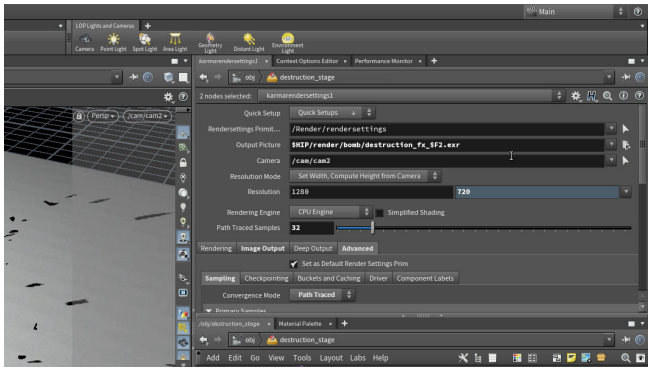


09 Scene View で**タンブル**し、爆発を上から見下ろす新しいカメラアングルにします。**LOP Lights and Cameras** シェルフで **Ctrl** キーを押しながら **Camera** ツールをクリックします。

Primitive Path を ***/cam/$OS*** に設定し、**名前** を ***cam2*** にします。



10 **Lock Camera to View** をオンにし、ビューポイントを微調整して目的のショットにします。210 と 240 の間のさまざまなポイントを確認して、準備ができたら **Lock Camera to View** オプションをオフにします。



11 ***cam2*** ノードを ***karmarendersettings2*** ノードの上に移動します。***usdrender_rop2*** ノードで、**Start** と **End** を **211** と **240** に変更し、Camera を ***/cameras/cam2*** にします。

1 つ目の ***karmarendersettings*** ノードを選択し、Camera が ***/cam/cam1*** に設定されていることを確認します。そうでない場合、デフォルトの ***camera1*** はシーン内にないので、ノードはレンダリングしません。

両方の ***karmarendersettings*** ノードを選択し、Output Picture を ***$HIP/render/bomb/destruction_fx_$F2.exr*** に変更します。

12 ***usdrender_rop1*** ノードを選択します。**Render to Disk** ボタンをクリックします。usdrender_rop2 でも繰り返します。

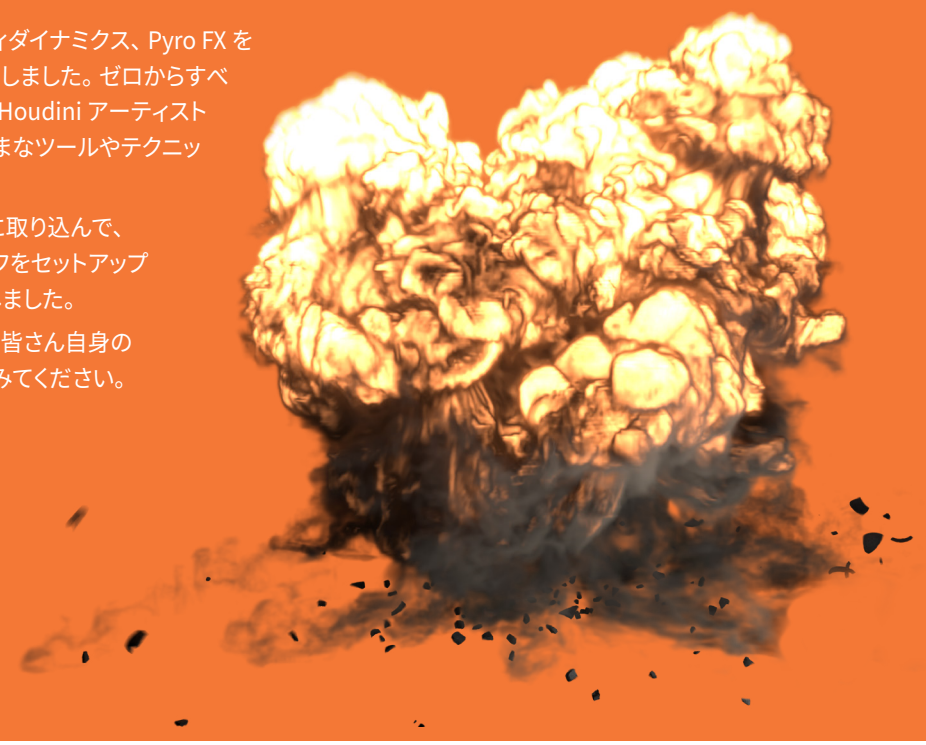
Render メニューに移動し、**MPlay > Load Desk Files** を選択します。***render/bomb*** ディレクトリに移動し、画像シーケンスを選択して、**Load** をクリックします。これにより、1 つのアニメーションとして画像が再生されます。

まとめ

パーティクル、リジッドボディダイナミクス、Pyro FX を使用して破壊ショットを作成しました。ゼロからすべてのプロジェクトを作成し、Houdini アーティストが日常的に使用するさまざまなツールやテクニックを体験しました。

また、作業内容を Solaris に取り込んで、USD を使用してシーングラフをセットアップし、Karma でレンダリングしました。

これらのスキルを活用して、皆さん自身の破壊 FX ショットを探求してみてください。



HOUDINI FOUNDATIONS

地形の生成

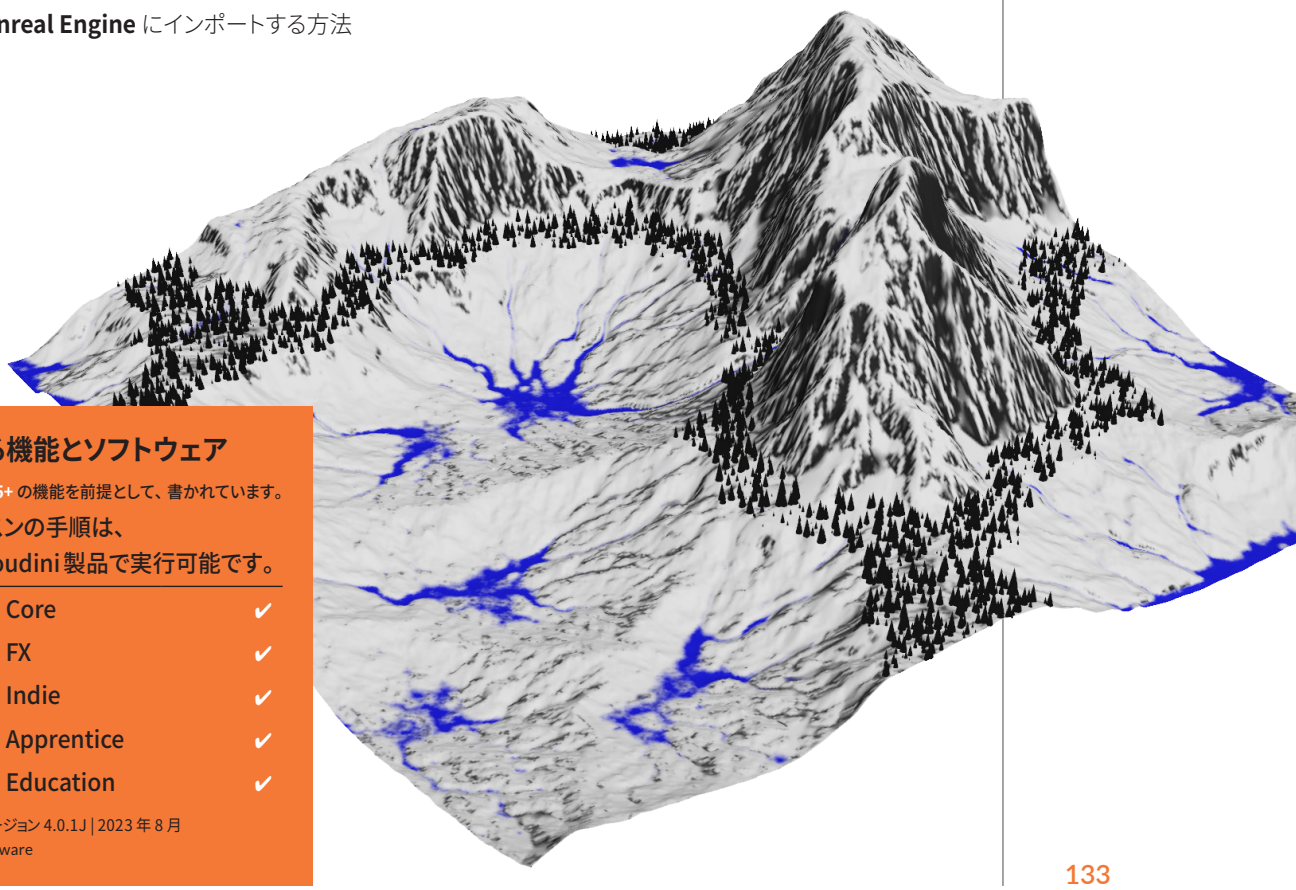
Houdini には、地形を生成したり、形状変更するための専用のツールセットが備わっています。これらのツールは、**Height Field** と呼ばれる 2D ボリュームを使って地形を表現します。このボリュームの各ボクセルには、そのグリッドポイントでの地形の高さが含まれています。Houdini のビューポートでは、2D Height Field を 3D サーフェスとして視覚化できます。また、Mask Field をセットアップして、地形の特定の部分に編集を集中させることもできます。このレッスンでは、パターン、ノイズ、侵食を使用して地形を構築し、その結果をゲームエンジンで使用できるようにエクスポートします。

レッスンの目標

Houdini の Height Field ツール を使用して地形を作成し、Unreal Engine または Unity に取り込みます。

学習内容

- **Height Field** を使用して地形を作成する方法
- **パターン**、**ノイズ**、**歪み**を追加する方法
- 地形の特徴に基づいて**マスク**を作成する方法
- Height Field に**ポイントをばら撒く**方法
- **Terrain Scattering** を使用してインスタンス化をセットアップする方法
- 地形をデジタルアセット (HDA) として**エクスポート**する方法
- HDA を **Unreal Engine** にインポートする方法



使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、以下の Houdini 製品で実行可能です。

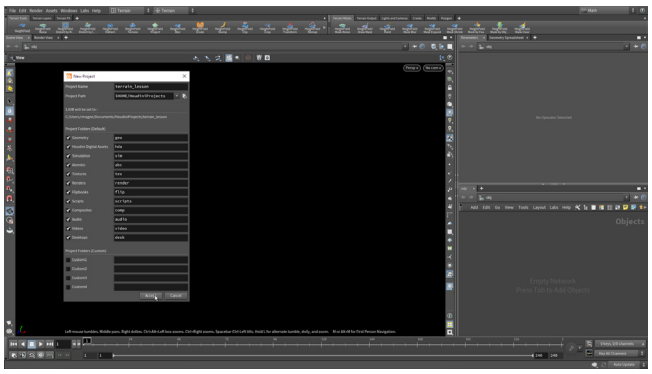
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

ドキュメントバージョン 4.0.1J | 2023 年 8 月

パート 1

Height Field による地形の形状変更

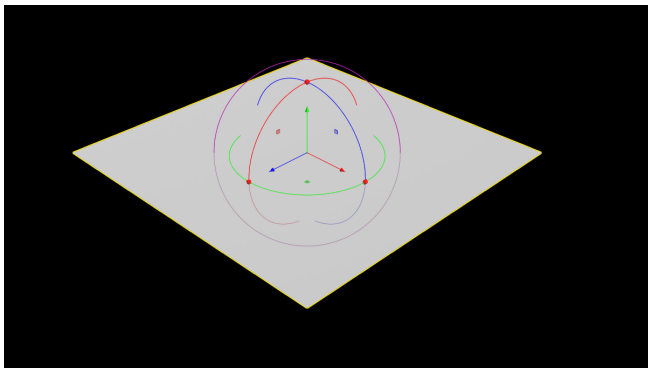
Houdini で地形を作成するには、Height Field を使用します。空の Height Field から始め、ノイズと歪みを加えて、景観の基本的なルックを定義します。ノード上でパラメータ値を微調整しながら、ディテールを重ねていきます。



01

デスクトップのセレクタで、**Terrain** を選択します。地形に特化したシェルフツールと Radial メニューが表示されます。**ディスプレイオプション**の一番上にある **Reference Plane** ボタンをオフにしたら、ビューポートで **D を押し**、**Background** タブの **Color Scheme** を **Dark** に設定します。

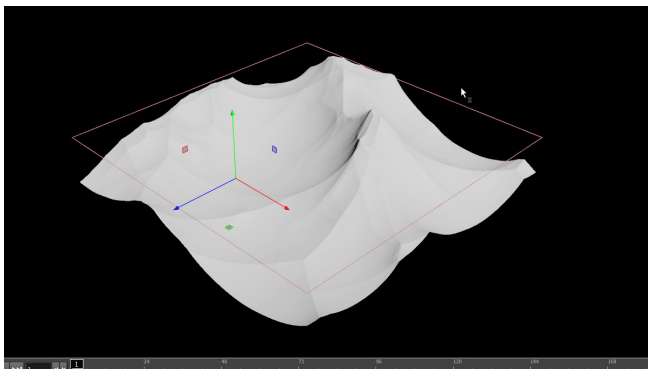
File > New Project を選択します。**Project Name** を ***terrain_lesson*** に設定し、**Accept** を押します。**File > Save As...** を選択すると、新しい ***terrain_lesson*** ディレクトリが表示されます。表示されない場合は、サイドバーの $JOB をクリックすると、そのディレクトリが表示されるはずです。ファイル名を ***terrain_01.hip*** に設定し、**Accept** をクリックして保存します。



02

Terrain ツールシェルフで、**Height Field** ツールをクリックします。Enter を押して原点に配置します。**スペースバー + H** を押し、Height Field 全体を表示します。

これにより、1000 x 1000、グリッド間隔 2 のグリッドが定義されます。



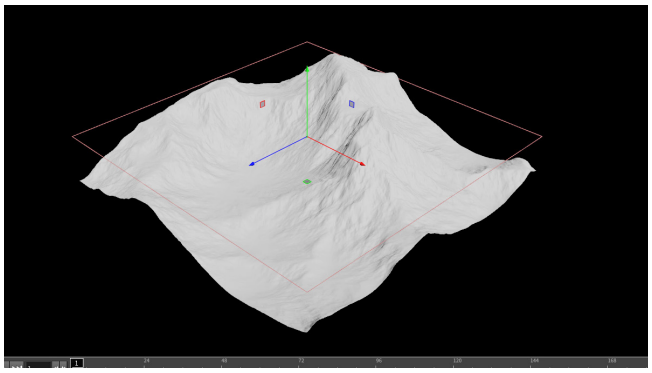
03

メインの Radial メニュー（**ホットキー C**）を使用して、**Deform > Noise** を選択します。次のように設定します。

- **Noise Type** を **Worley Cellular F1** にする
- **Amplitude** を **360** にする
- **Offset** を **20, 0, 300** にする

地形は、このようなノイズから始めるのがお勧めです。

メインの Radial メニューを使用して、**Deform > Blur** を選択し、**Heightfield Blur** ツールにアクセスします。**Radius** を **20** に設定します。エッジがソフトになり、風化作用の影響が示されます。



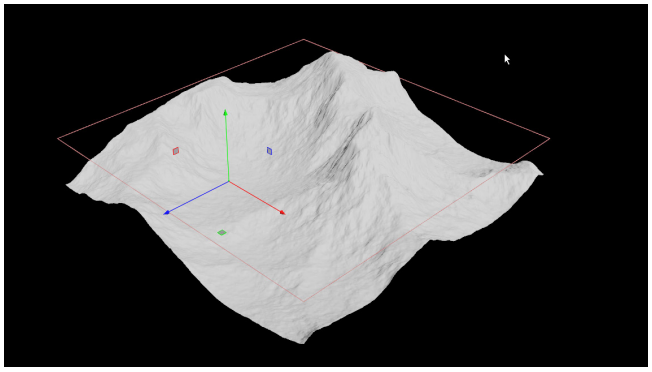
04

Radial メニューを使用して **Deform > Distort** を選択します。次のように設定します。

- **Amplitude** を **40** にする
- **Element Size** を **220** にする

このノードは、ノイズフィールドを通して既存の値を移流させ、それらを動かします。ここで提案されたパラメータ値をそのまま使用しても、独自に試して好みのルック（外観）にしてもかまいません。

Houdini のプロシージャルアプローチなら、後で戻ってパラメータ値を変更し、設定によって結果がどう変化するかを確認することができます。



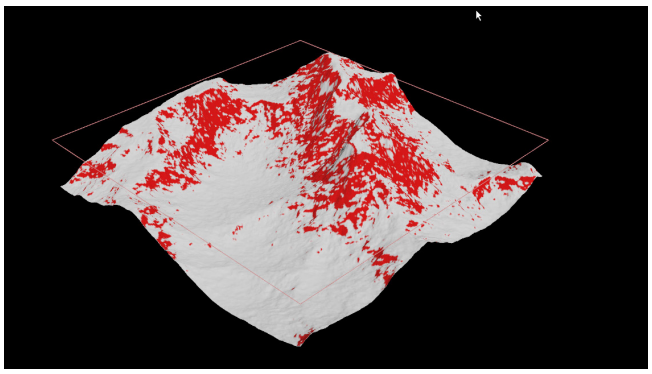
05

Radial メニューを使用して **Deform > Noise** を選択します。次のように設定します。

- **Amplitude** を **10** にする
- **Element Size** を **20** にする

ネットワークビューで最後の 4 個のノードを選択したら、**Network box** アイコンをクリックしてネットワークボックスを追加します。エッジを調整してボックスの形状を整えたら、上部のバーをクリックして ***Shape the Terrain*** と名前を付けます。

ネットワークボックスは、ネットワークを読みやすくするため、他のアーティストとファイルを共有する場合に特に便利です。



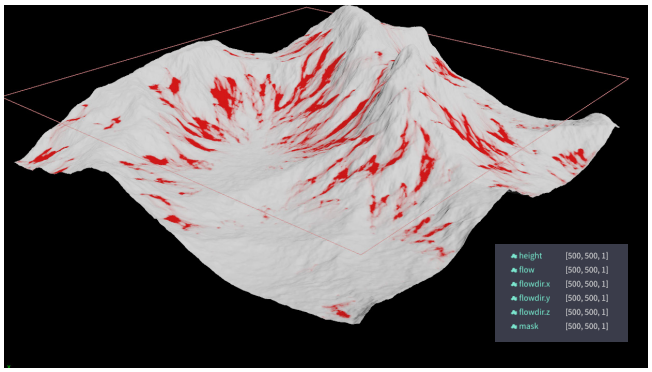
06

Radial メニューを使用して **Mask > Mask by Feature** を選択します。次のように設定します。

- **Min Slope Angle** を **35** にする
- **Max Slope Angle** を約 **60** にする

これで、山の側面の領域に集中できるようになります。

このノードに送られる地形の形状を後から変更したら、それに応じてマスクも更新されます。

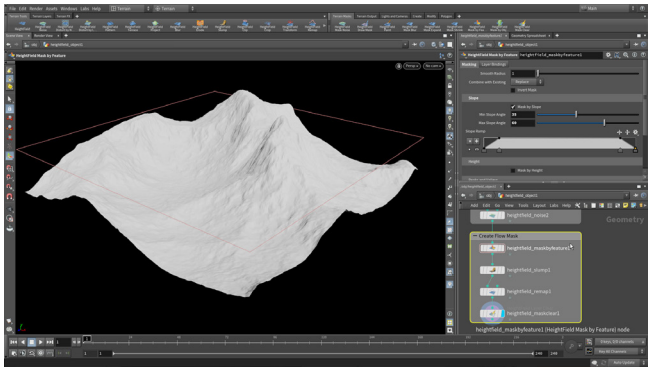


07

Radial メニューで **Erode > Slump** を選択します。**Spread Iterations** を **75** に設定します。

heightfield_slump ノードが作成するのは、不安定に重なった岩片が、安定した状態に移行しようとする際に起こる崩壊による浸食です。マスクレイヤーに影響を与え、**Flow** および **Flow Direction** レイヤーを出力します。

このノードを **MMB クリック**すると、どのレイヤーが作成されているかを確認できます。比較するには、チェーンの前のノードを **MMB クリック**します。



08

Terrain ツールシェルフで、**Heightfield Remap** ツールを選択します。**Layer to Remap** を **Flow** に設定したら、**Compute Range** ボタンをクリックします。**Output Max** を **1** に設定して、これらの値を正規化します。Radial メニューを使用して **Mask > Clear Mask** を選択します。マスクレイヤーがクリアされ、セットアップにより多くのレイヤーを作成できるようになります。

ネットワークビューで最後の 4 個のノードを選択したら、**Network box** アイコンをクリックしてネットワークボックスを追加します。エッジを調整してボックスの形状を整えたら、上部のバーをクリックして ***Create Flow Mask*** と名前を付けます。作業内容を**保存**します。

HEIGHT LAYER

ジオメトリネットワークに渡すデータには、複数の Height Field を含めることができます。Houdini の Terrain ツールの用語では、これらのデータのことを Height Layer と呼びます。例えば、ツールが 1 つの Height Layer で岩盤を表現し、別の Height Layer でその岩盤に重なった緩い土壌を表現することができます。デフォルトの Height Layer の名前は height です。

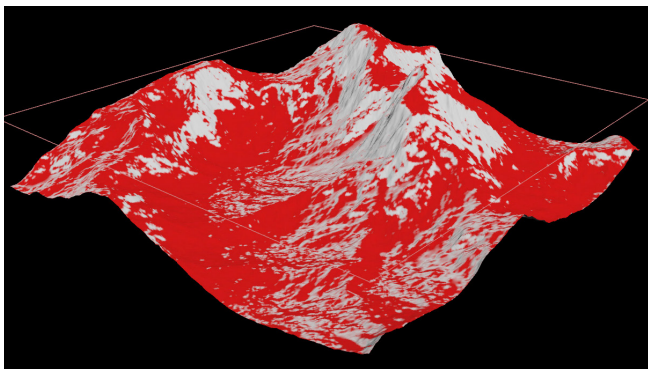
各ボクセルには、マスクレイヤーと呼ばれる「選択性」の値が含まれています。ほとんどの Terrain ノードは、2 番目の入力でマスクレイヤーを受け取り、マスクによってそのノードが修正する地形の部分を制御することができます。デフォルトのマスクレイヤーの名前は mask で、3D サーフェス上に赤く表示されます。

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]

パート 2

マスクレイヤーの追加と視覚化

地形にレイヤーをセットアップするには、まずマスクを設定し、その情報を特定のレイヤーにコピーします。この操作を何回か行うことで、さらにレイヤーを追加できます。その後これらのレイヤーを使用して、地形の重要な特徴を視覚化します。



01

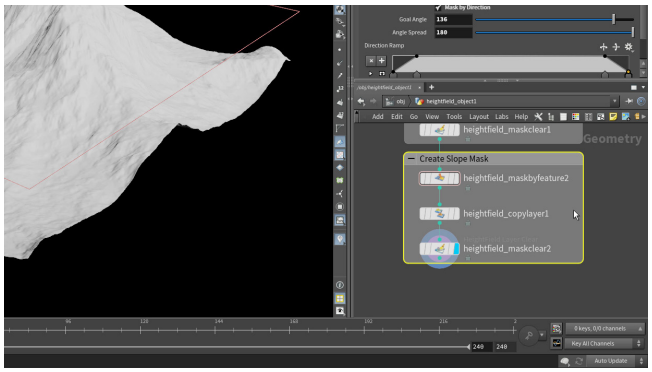
Radial メニューで **Mask > Mask by Feature** を選択します。**Mask by Slope** で次のように設定します。

- **Min Slope Angle** を **0** にする
- **Max Slope Angle** を約 **45** にする

Mask By Direction をオンにして、次のように設定します。

- **Goal Angle** を約 **136** にする
- **Angle Spread** を **180** にする

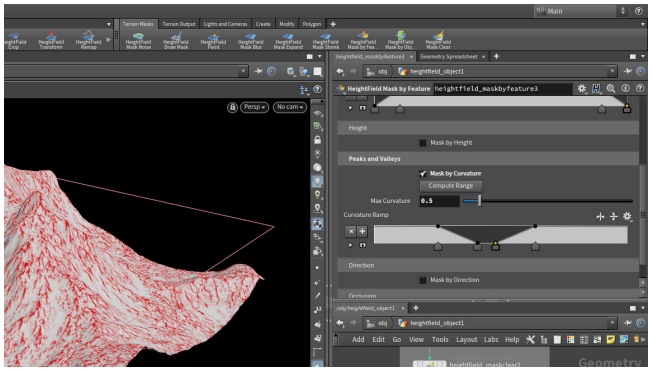
この設定は、谷を含む広い範囲の地形をカバーします。



02

Radial メニューで **Layer > Copy Layer** を選択します。**Source** を Mask のままにして、**Destination** を ***slope*** に設定します。マスクを新しいレイヤーにコピーすることで、マスクをクリアして他のタスクに使えるようになります。

Radial メニューで **Mask > Clear Mask** を選択します。マスクレイヤーが再度クリアされ、セットアップにより多くのレイヤーを作成できるようになります。ネットワークボックスを追加してノードを整理し、そのボックスに ***Create Slope Mask*** と名前を付けます。



03

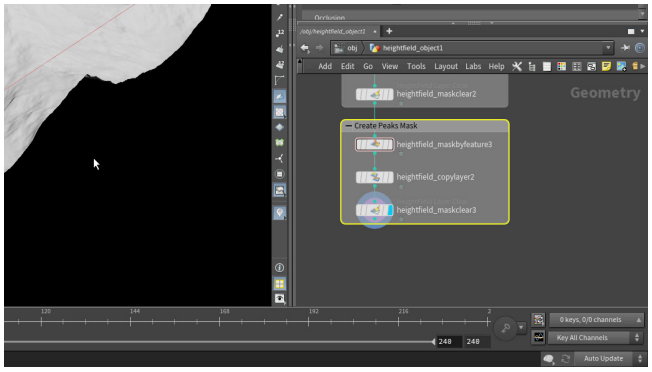
メインの Radial メニューを使用して **Mask > Mask by Feature** を選択します。**Mask by Slope** で次のように設定します。

- **Min Slope Angle** を **0** にする
- **Max Slope Angle** を約 **70** にする

Mask By Curvature をオンにして、次のように設定します。

- **Max Curvature** を **0.5** にする

Curvature Ramp のポイントを中心に向かって動かし、地形のピークを見つけます。これで非常に細かいマスクができるので、景観の主な特徴すべてのピークを特定できます。



04

Radial メニューで **Layer > Copy Layer** を選択します。**Source** を Mask のままにして、**Destination** を ***peaks*** に設定します。次に、**Mask > Clear Mask** を選択します。再度マスクレイヤーをクリアします。

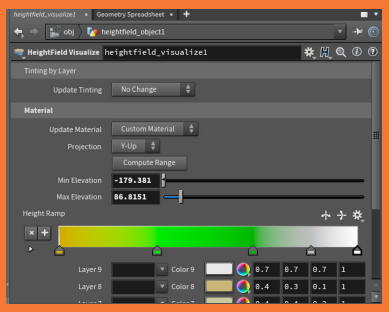
これで、マスクから 3 つのレイヤーができました。ネットワークボックスを追加してノードを整理し、ボックスに ***Create Peak Mask*** と名前を付けます。

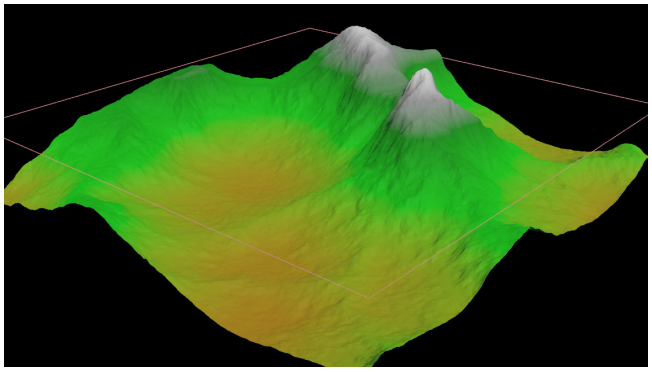
次の手順では、これらを使用して地形を視覚化します。

Height Field の視覚化

Height Field の視覚化は、地形全体の高さに割り当てられたランプから始めます。まず、ランプが相当する**範囲を計算**し、その後ランプでカラーを調整して、景観を視覚化します。

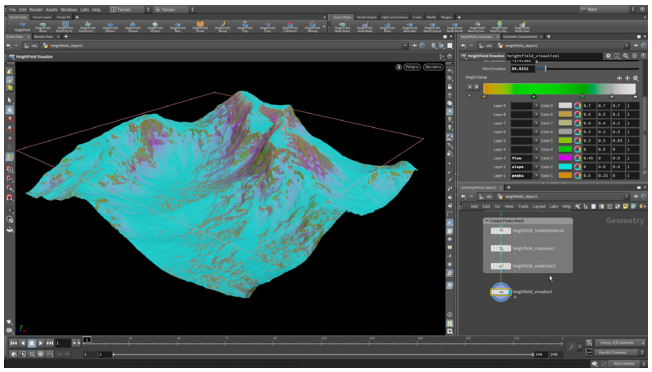
その後、ランプ上の各種レイヤーに色を追加します。これで、よりリッチなルックのシーンにできます。





05 Radial メニューで **Visualize > Heightfield Visualize** を選択します。**Compute Range** ボタンをクリックして、ビジュアライゼーションを現在の Height Field の範囲に揃えます。

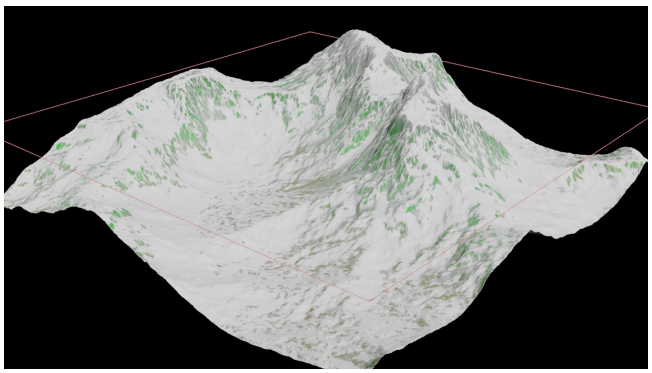
これにより、地形の最下部から最上部までの傾斜のビジュアライザが設定されます。3D ビューで傾斜がどのように見えるかを確認することができ、山の頂上は白でハイライトされます。



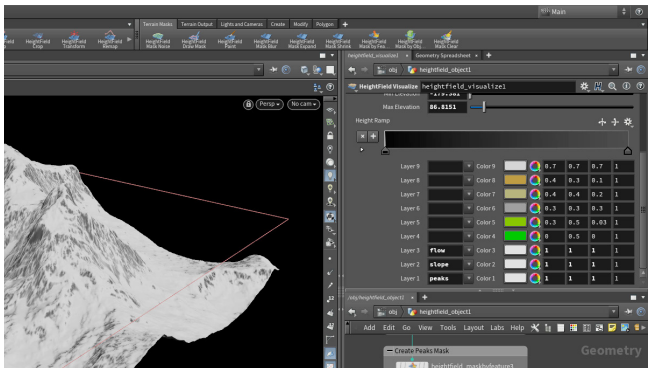
06 ランプウィジェットで、次のように設定します。

- **Layer 1** を ***peaks*** にする
- **Layer 2** を ***slope*** にする
- **Layer 3** を ***flow*** にする

マスクを使って作成した 3 つのレイヤーを使用して、3D ビューにデフォルトのカラーが表示されます。これらを使って地形のルックを定義していきます。



07 これら 3 つのレイヤーすべてを**白** (1, 1, 1) に設定します。すると雪で覆われたようなルックになります。これらのレイヤーを使ってさまざまな特徴を表現できますが、この山は、雪のルックを強調します。



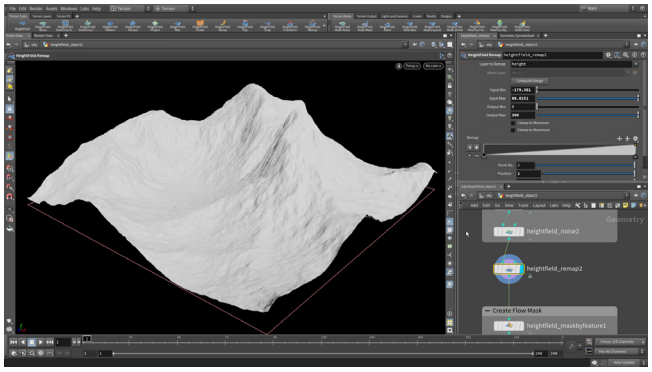
08 **Height Ramp** で、2 つを除くすべてのマーカーを選択して削除します。右側のマーカーを黒、左側のマーカーをダークグレーに設定します。これにより、雪のレイヤーの下に暗いルックが作成され、暗い領域が視覚的に目立って見えるようになります。

作業内容を**保存**します。

パート 3

地形の再マップと浸食

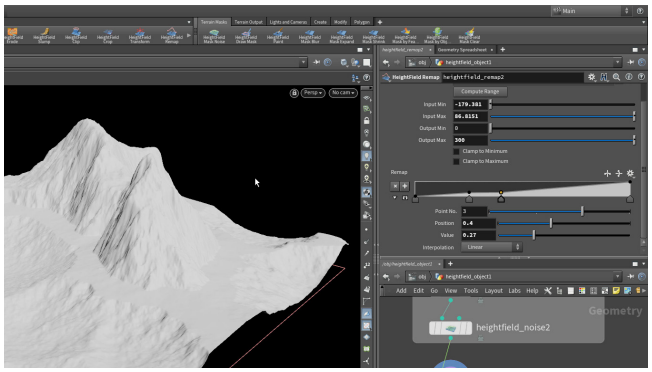
現在、高さマップには 0 より下の領域と、0 以上の領域があります。Remap ノードを使用して範囲を変更してから、
そのノードのランプを使用して山の周りに細長い台地を追加します。その後、浸食させて、地形に新しいレイヤーを追加します。



01 Shape by Terrain ネットワークボックスの最後にある **Heightfield Noise** ノードを見つけます。**Display フラグ**をオンにし、クリックしてこのノードを選択します。

Terrain ツールシェルフで、**Height Remap** ツールをクリックします。**Compute Range** ボタンをクリックします。高さを再構成するために、次のように設定します。

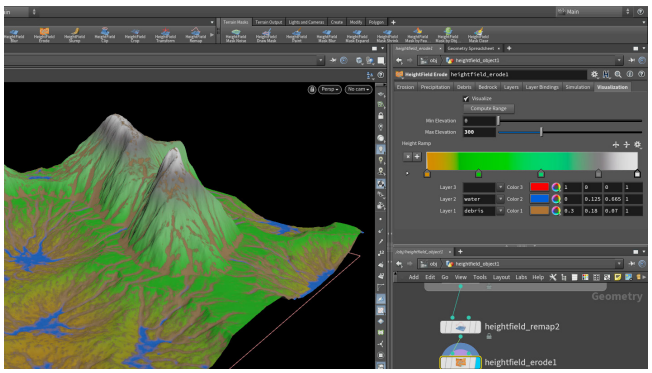
- **Output Min** を **0** にする
- **Output Max** を **300** にする



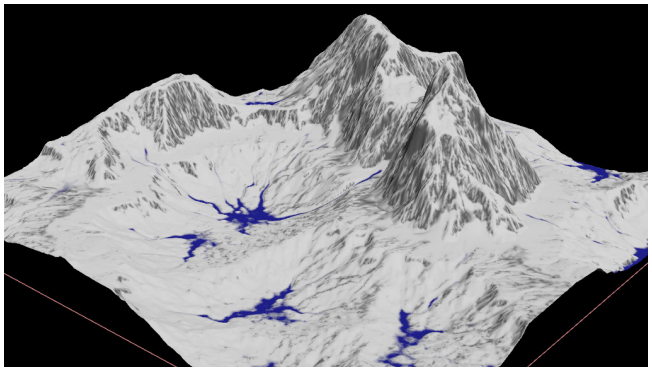
02 **Remap** ランプを使用して、中央にポイントを追加したら、さらに次のようにポイントを追加して調整します。

- **Point 2** を追加し、**Position** を **0.25**、**Value** を **0.25** に設定する
- **Point 3** を追加し、**Position** を **0.4**、**Value** を **0.27** に設定する

これで、山の基部に沿って細長い台地が作成されます。



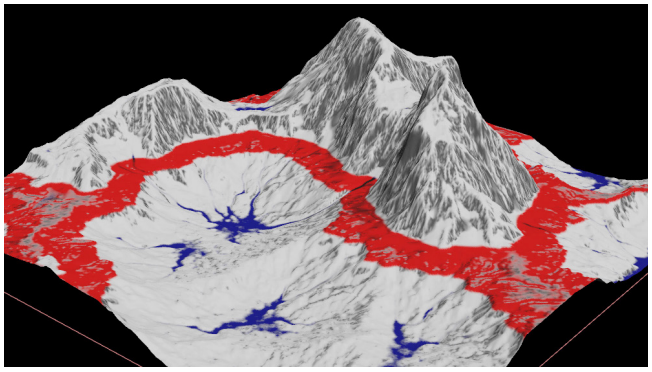
03 **Erode > Erode** を選択します。**Visualization** タブをクリックし、**Compute Range** ボタンをクリックします。**Play** を押して、地形の浸食を確認します。フレーム **15** あたりで **Stop** を押します。



04 **Display フラグ**をチェーンの最後にある **Heightfield Visualization** ノードに戻します。Erode ノードに由来するレイヤーである **Layer 3** を ***water*** に設定し、**カラー**を**青**に変更します。これにより、ビジュアライゼーションでこれらの領域がはっきり見えるようになります。

パート 4
地形にポイントをばら撒く

木と岩を追加するには、新しい台地の領域をマスクしてから、このマスクを使用する地形へのばら撒きをセットアップします。
これらのばら撒かれたポイントは、木を表す、インスタンス化された円錐をコピーするのに使用されます。
これらは後ほど Unreal で置き換えます。



01

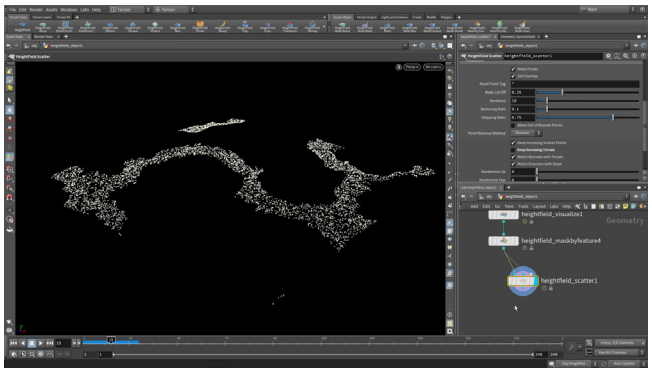
Radial メニューを使用して **Mask > Mask by Feature** を選択します。次のように設定します。

- **Min Slope Angle** を **0** にする
- **Max Slope Angle** を **50** にする

Mask by Height をオンにして、次のように設定します。

- **Min Height** を **70** にする
- **Max Height** を **85** にする

これにより、***Remap*** ノードで作成した細長い台地がハイライトされます。ハイライトされない場合は、マスクでこの領域が分離されるまで、これらの値を調整してください。

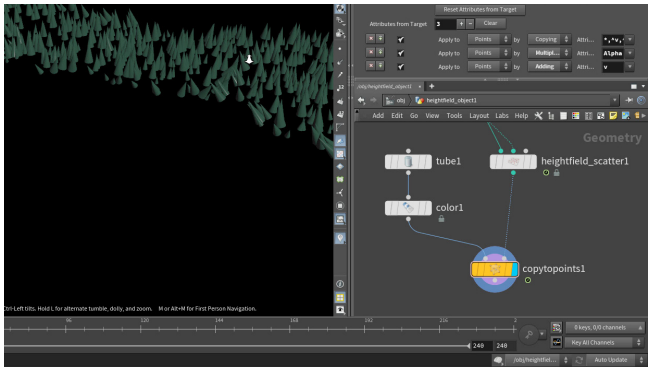


02

ネットワークエディタで、***maskbyfeature*** ノードの出力を **RMB クリック**し、**scatter...** と入力していきます。

Heightfield Scatter を配置して、その **Display フラグ**をオンにします。***maskbyfeature*** ノードの出力を ***scatter*** ノードの 2 番目の入力に接続して、入力マスクで定義した領域にポイントを制限します。

Coverage を **0.05** に設定します。**Keep Incoming Terrain** オプションを**オフ**にします。



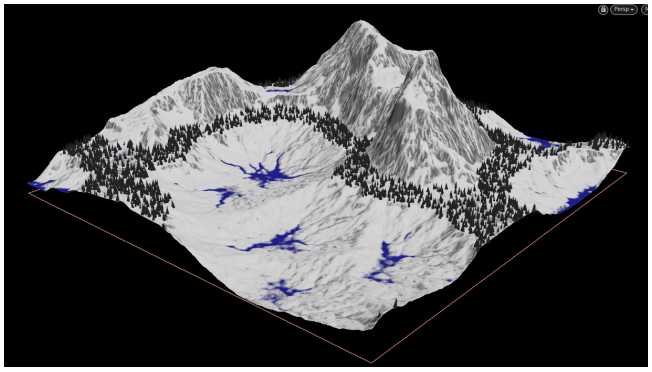
03

ネットワークエディタで、**Tab** を押して **Copy to Points** と入力していき、クリックしてそのノードを追加します。**Pack and Instance** チェックボックスをクリックしてオンにします。**Heightfield Scatter** を **Copy** の 2 番目の入力に接続して、Display フラグを設定します。

ネットワークエディタで、**tube** ノードを追加し、***copytopoints*** ノードの 1 番目の入力に接続します。次のように設定します。

- **Radius** を **0, 2** に、**Radius Scale** を **1** にする
- **Height** を **10** に、**Center** を **0, 5, 0** にする

tube の後に **color** ノードを追加して、木を**緑**にします。**Merge** ノードを追加して、それに ***Heightfield Visualize*** と ***copytopoints*** を接続します。



04

heightfield_scatter ノードに戻り、**Match Normals with Terrain** と **Match Direction with Slope** をオフにします。これで、すべての木が上を向くようになります。

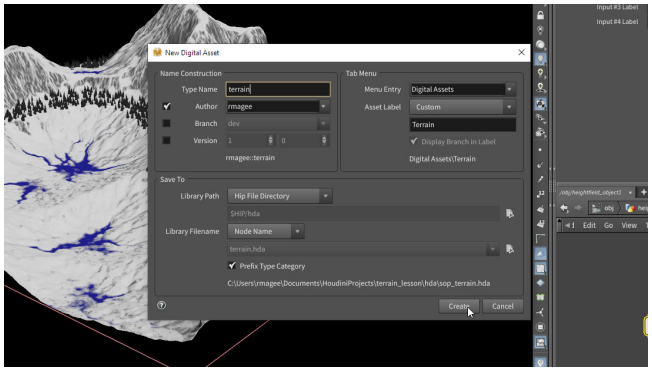
ここで **Random Up** を **10** に変更して向きに変化を付け、**Randomize Yaw** を **20** に設定します。この効果は分かりにくいですが、後で木を置き換えると、ランダムに回転しているのが確認できます。

木のスケールは **Variability** でコントロールします。**Range** を **1, 2** に変更して、これら 2 つの値の間でランダムにスケールするようにします。

パート 5

Unreal で地形を開く

Unreal Engine または Unity のようなゲームエンジンに地形を取り込むには、まず Houdini デジタルアセットを作成します。Houdini Engine プラグインを適切にセットアップしたら、このアセットをゲームエディタにロードして、コピーした木のスタンドインをインスタンス化されたオブジェクトとしてインポートすることができます。地形を Unreal Engine にインポートすると、Height Field はランドスケープとして認識されます。Houdini Engine プラグインを使用して、Unity にアセットをインポートすることもできます。

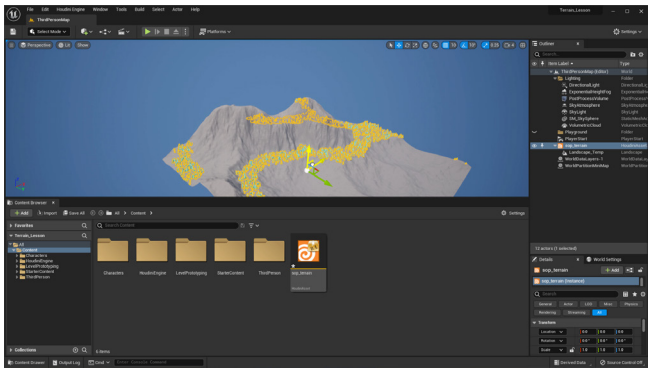


01

ネットワークエディタですべてのノードを選択します。Create **Subnet from Selected ボタン**をクリックします。新しいサブネットを **RMB クリック**して、**Digital Asset > Create New** を選択します。

Type Name を ***terrain*** に設定し、**Branch** と **Version** をオフにします。**Library Path** を ***HIP File Directory*** に、**Library Filename** を ***Node Name*** に設定します。

Create をクリックします。**Edit Type Properties** ウィンドウが開きます。**Accept** をクリックして、このウィンドウを閉じます。



02

Unreal Engine を起動したら、メインパネルで **New Project** タブをクリックして、**Third Person** テンプレートを選択します。**Create Project** をクリックします。開いたら、地形を妨げないようデフォルトのジオメトリを削除します。

Content Browser で、**Import to Game** をクリックして ***terrain.hda*** アセットファイルを見つけます。そのアセットを **Content Browser** から 3D ワークスペースにドラッグします。**ThirdPersonCharacter** の **Translate Z** を約 **10,000** に設定したら、**Play** を押して地形の周りを歩きます。

03

Houdini Instanced Inputs セクションに移動して、***terrain1_1*** を展開します。この円錐のインスタンスを、Unreal 内のコンテンツで置き換えます。

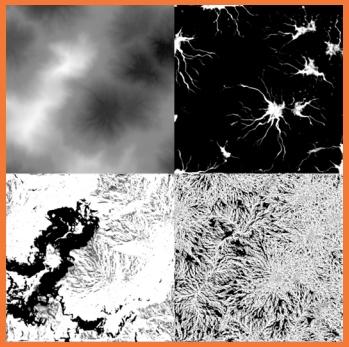
Content Browser で、**StarterContent > Props** を開きます。***SM_Bush*** プロップを **Houdini Instanced Input** 上にドラッグします。**Scale Offset** を **5, 5, 5** に設定します。ジオメトリがポイントにインスタンス化され、円錐のようにランダムにスケールおよび回転されます。

アウトライナーで、**terrain** 下の **Landscape** ノードを選択します。**Landscape** マテリアルの横にあるメニューをクリックして、草のマテリアルを見つけます。**Play** を押して、地形を確認します。

まとめ

地形レイヤーを使用してテクスチャマップを作成し、それを使って景観のルックを定義することもできます。これを行うには、**Heightfield Output** ノードを使用します。これらのチャンネルを使用して、景観の特徴を参照したマテリアルを Unreal Engine で構築できます。

Houdini の地形の機能について、概要を把握しておくことをお勧めします。ゲーム用に景観を作成し、岩や木などのディテールを追加する際の可能性が大きく広がります。



CHARACTER FOUNDATIONS

KINEFX リギング | FUR DUDE

このレッスンでは、2 足歩行キャラクタ、ファー・デュード (Fur Dude) のリギングとアニメーションを行い、その後ファーを追加します。既存のジオメトリから始め、スケルトンを描画し、ジオメトリをキャプチャしてから、アニメーションリグのリグコントロールを構築します。その後、歩行サイクルのキーフレームを設定し、サーフェスにファーを追加します。

このレッスンでは、Houdini の新しい SOP ベースのプロシージャルリギングツール **KineFX** を使用します。主にリターゲットワークフローに使用するツールですが、キャラクタやクリーチャのリギングに使えるツールも含まれています。ただし、ツールはどれも発展の過程にあります。このレッスンで体験していただくのは、現時点で可能なことです。今後のリリースで、KineFX やアニメーションのワークフローはさらに拡張され、洗練されていく予定です。

レッスンの目標

クリーチャ「ファー・デュード」のリギングとアニメーション、ファーの追加を行います。

学習内容

- **KineFX** ジョイントを使用してスケルトンを作成する方法
- 変形およびリジッドジオメトリをスケルトンにキャプチャする方法
- キャプチャリグをデジタルアセットにラップする方法
- コントロールを追加して、アニメーションリグを構築する方法
- 歩行サイクルをアニメートする方法
- クリーチャにファーを追加する方法
- Solaris と Karma を使用してレンダリングする方法

使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、以下の Houdini 製品で実行可能です。

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

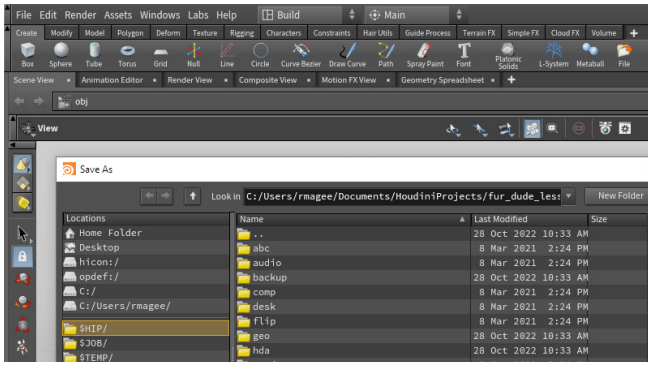
ドキュメントバージョン 4.0.1J | 2023 年 8 月

パート 1
スケルトンの描画

シーンファイルを開き、ファー・デュードのジオメトリを確認してから、Skeleton ツールを使用してジョイントを配置します。このツールは、ジョイントを作成して名前を付けたり、アニメートしたいキャラクタに合わせて調整するために使えます。

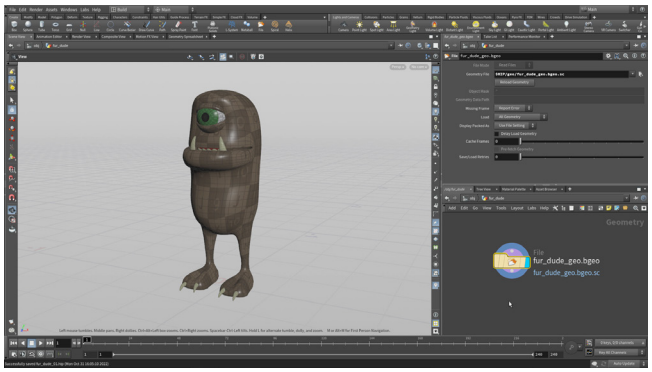
プロジェクトファイル

SideFX.com のファー・デュードのチュートリアルページ（このドキュメントを入手した場所です）から、***furdude_lesson_start*** ディレクトリをダウンロードします。名前を ***furdude_lesson*** に変更し、**Houdini Projects** ディレクトリに配置してください。



01 **File > Set Project** を選択します。先ほどダウンロードした ***furdude_lesson*** ディレクトリに移動し、**Accept** を押します。これにより、先ほどコピーしたプロジェクトディレクトリとそのサブフォルダに、このショットに関連するファイルがすべて配置されるようになります。

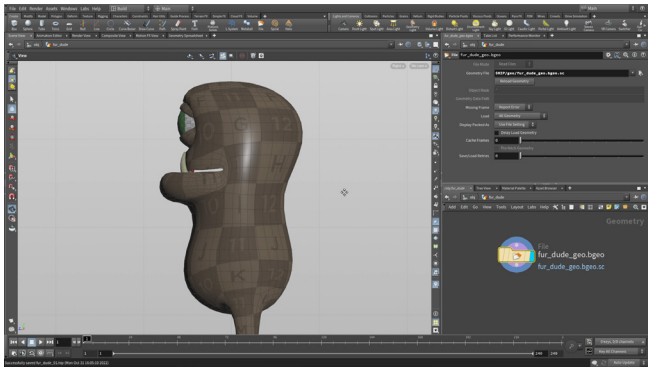
File > Open を選択すると、新しい ***furdude_lesson*** ディレクトリが表示されます。***furdude_start.hip*** という名前のファイルを開きましょう。**File Save As...** を選択し、***furdude_01.hip*** にファイル名を変更します。**Accept** をクリックして保存します。こうしておくと、後でもう一度レッスンをやりたくなったときに、手つかずのスタートファイルに戻れます。



02 シーンを開くと、***fur_dude_rig*** という名前のオブジェクトが 1 つ表示されます。これから、**KineFX** ツールセットを使用して作成したボーンに、このジオメトリをキャプチャしていきます。

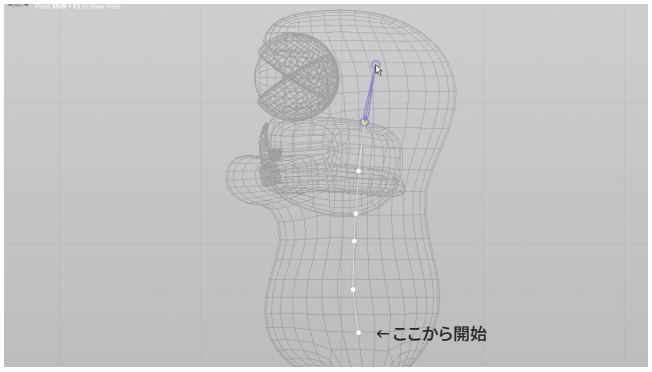
ノードを**ダブルクリック**して、ジオメトリレベルに入ります。このレッスンプロジェクトでは、ディスクから ***fur_dude_geo.bgeo*** ファイルをインポートしている **File** ノードを確認できます。

このジオメトリには、プリミティブカラーやグループなどの情報が保存されています。情報を確認するには、ノードを **MMB クリック**して、リストされたアトリビュートやグループを表示します。これらのグループは、後でジオメトリをキャプチャする際に使用します。



03 Scene View にカーソルを移動し、**スペースバー + B** を押して 4 画面ビューにします。右上のアイコンから、**Link Ortho Views** をオンにします。こうすると、Top、Front、Right のビューをすべて同時にパン、ズームできるようになります。

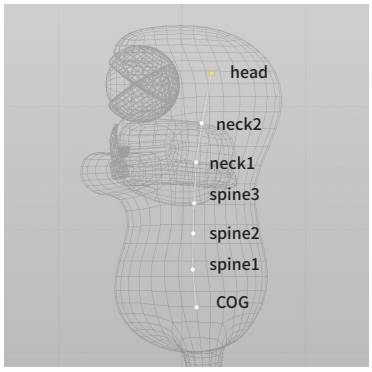
Right ビューにカーソルを移動し、再度**スペースバー + B** を押します。ジョイントの描画には、このビューがやりやすいはずです。

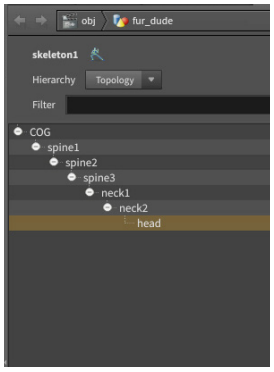


04 ネットワークビューで **Tab > Skeleton** を選択し、File ノードの横に新しいノードを配置します。そのノードに **Display フラグ**を設定します。File ノードに **Template フラグ**を設定して、skeleton ノードの作業時に、ジオメトリがグレーのワイヤーフレームとして表示されるようにします。

Handle ツールが選択されていることを確認します。上部のバーで、**Joint Placement** を **Freehand** に設定します。こうすると、ジオメトリとは関係なく、コンストラクション平面での描画が可能になります。脚のすぐ上に最初のジョイントを配置したら、図のように、上方向に 6 つのジョイントを配置します。

MMB クリックして、ジョイントの描画を終了します。

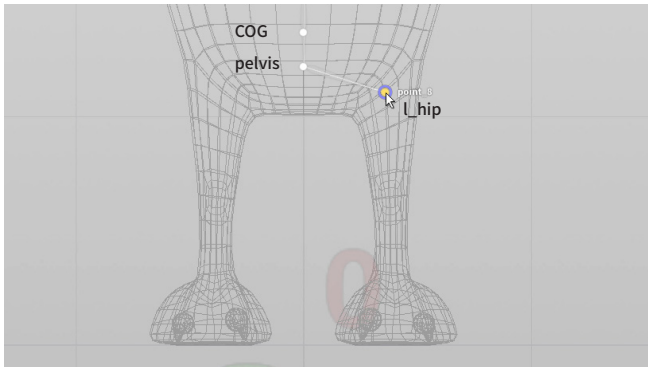




05

上部のバーで、**Mode** を **Modify** に設定します。これで、ジョイントを編集できるようになります。最初のジョイントをクリックして、上部のバーで **Name** を ***COG*** に設定します。

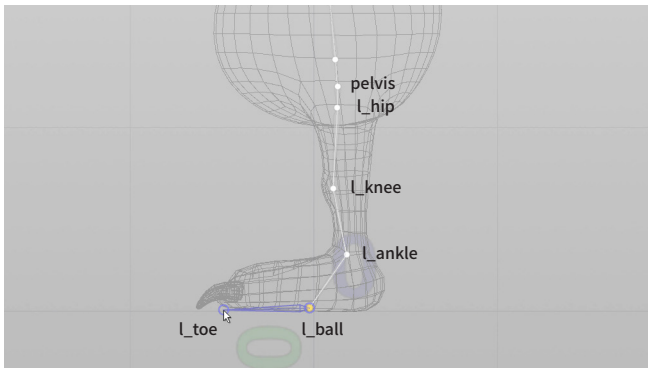
パラメータエディタのタブ領域で、**+（プラス）記号**をクリックします。**New Pane Tab Type > Animation > Rig Tree** と選択します。すると、スケルトンジョイントを表示したペインが表示されます。2 つ目のジョイントをダブルクリックして、***spine1*** と名前を付けます。Scene View または **Rig Tree** を使用して、残りのジョイントに図のように名前を付けます。



06

上部のバーで、**Mode** を **Create** に戻します。デフォルトでは、選択したジョイントの端から描画を始めることになります。**MMB クリック**して、その動作を停止します。Right ビューで ***COG*** ジョイントをクリックし、その下に ***pelvis*** ジョイントを描画します。

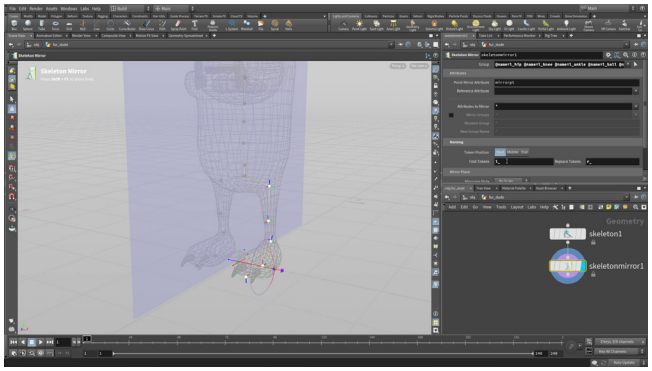
Scene View で、**スペースバー + B** を押して 4 面ビューに戻ります。Front ビューで、キャラクタの左側に ***hip*** ジョイントを描画します。



07

Right ビューに戻り、脚の最後の 4 つのジョイントを図のように描画します。

Modify モードに戻し、**Rig Tree** を使用してジョイントの名前を変更します。または、ジョイントを選択し、上部のバーで名前を変更することもできます。***pelvis*** の後、すべてのジョイントに「***l_***」の接頭辞を付けるのは、これらのジョイントを左脚に使用するためです。



08

Scene View で、**Tab > Skeleton Mirror** を選択します。このコマンドは、すべてのジョイントのミラーコピーを作成します。パラメータエディタに移動して、**Group** の横の矢印をクリックします。脚のジョイントのみを選択し、Enter を押します。脚のみがミラー化されました。

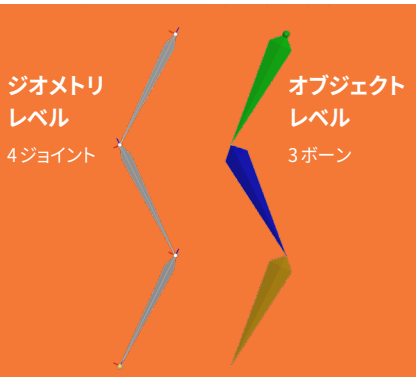
Naming で、**Find Tokens** を ***l_***、**Replace Tokens** を ***r_*** に設定します。これで、右脚のジョイントに適切な名前（接頭辞 r_）が付けられます。

File > Save を選択して、ここまでの作業を保存します。

KINEFX とオブジェクトレベルのリギング

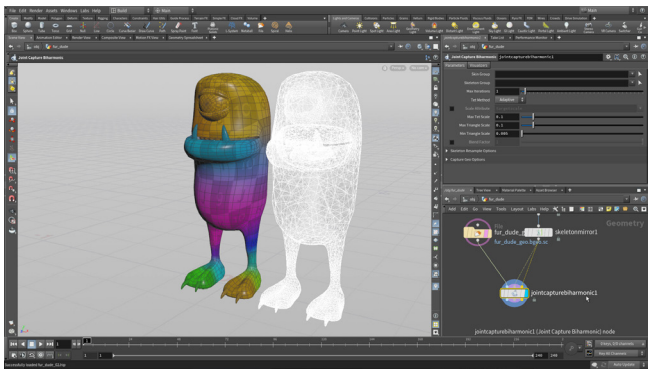
Houdini の KineFX ツールは、ジオメトリ（SOP）レベルでのジョイントベースのワークフローを提供します。Houdini の他のキャラクタワークフローはボーンベースで、この場合、作業は主にオブジェクト（OBJ）レベルで行うことになります。

KineFX ワークフローでは、ジョイントは基本的にカーブ上のポイントにすぎません。このように扱うことで、SOP レベルのツールでリグを操作する可能性が大きく開けました。ここでは、キャラクタやクリーチャのリギング専用に設計されたツールについて学習します。



パート 2
ジオメトリのキャプチャ

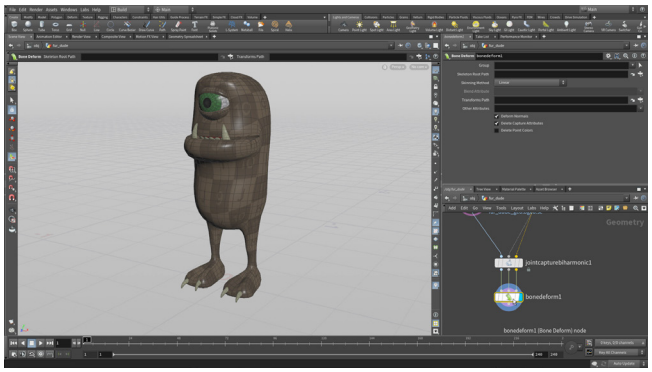
キャラクタのリギングには、ジオメトリをスケルトンジョイントにキャプチャする操作が含まれます。このようにすることではじめて、ジョイントの回転によってジオメトリが変形したり曲がるようになります。Houdini は、最初のキャプチャで素晴らしい結果を得られる Biharmonic（重調和）キャプチャ手法を使用しており、リグをすぐにテストできます。その後、キャプチャウェイトをペイントして結果を微調整し、キャラクタを動かせるようにします。



01 **スペースバー + B** を使用して、Scene View をパースビューに変更します。ネットワークビューを少し大きくして、作業スペースを確保します。

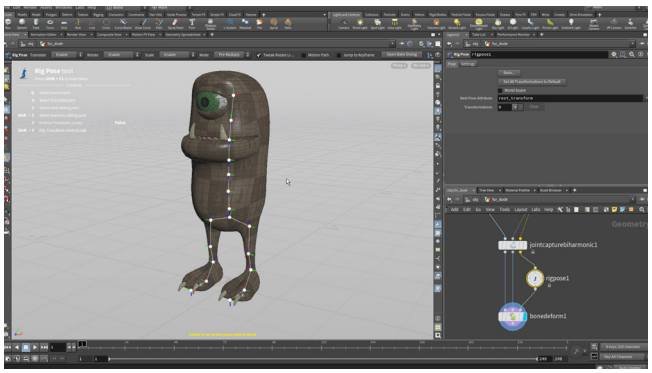
ネットワークビューで **Tab > Joint Capture Biharmonic** を選択し、このノードを skeleton ノードの下に配置します。***fur_dude_geo*** を ***jointcapturebiharmonic*** ノードの **1 つ目の入力**に接続します。***skeletonmirror*** ノードを ***jointcapturebiharmonic*** ノードの **2 つ目と 3 つ目の入力**に接続し、**Display フラグ**を設定します。

ジオメトリにキャプチャウェイトが表示されるようになりました。後でこれらを微調整およびペイントして、ジオメトリの変形をセットアップします。

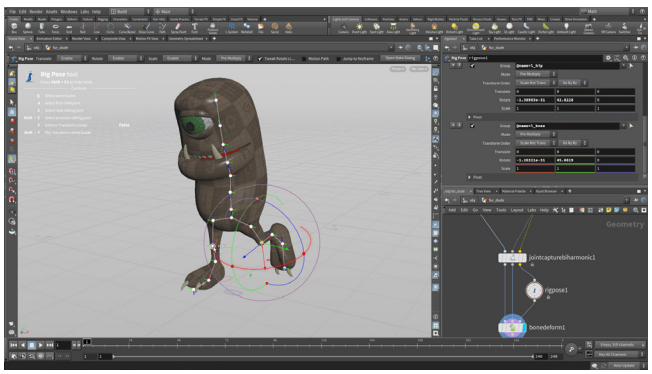


02 ネットワークビューで、**Tab > Bone Deform** を選択します。その **3 つの入力**に、***jointcapturebiharmonic*** の **3 つの出力**を接続します。***bonedeform*** に **Display フラグ**を設定します。

このレッスンでは、おそらく変形する必要のない歯、つめ、目をキャプチャします。これらは後ほど分離し、別の方法でキャプチャしていきます。



03 次に、**Tab > Rig Pose** を選択してノードを配置します。それを、***jointcapturebiharmonic*** と ***bonedeform*** の 3 つ目の入力を接続している 3 つ目のライン上に移動して、チェーンに追加します。ここでリグにアニメーションを付けます。rigpose を使用してジョイントを回転させたり、キーフレームを設定することができます。

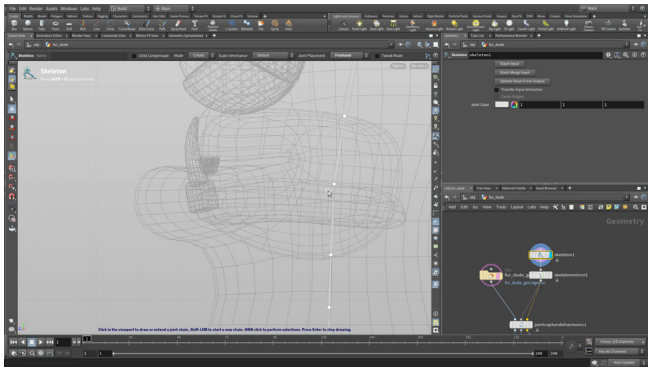


04 ***rigpose*** ノードを選択して、Scene View で **Handle** ツールがアクティブなことを確認します。さまざまなジョイントを**選択**および**回転**して、変形をテストします。これは後でリセットできるので、自由に試してください。

パート 3

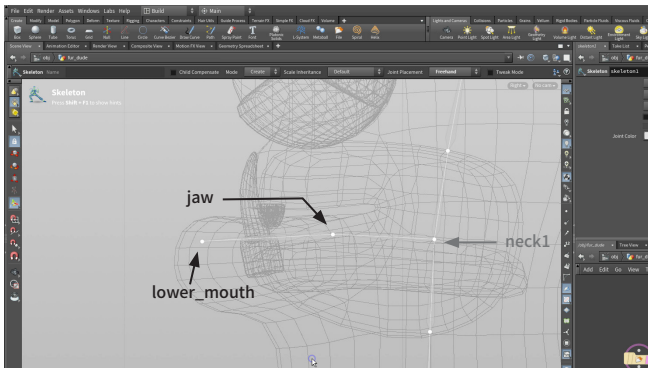
ボーンの追加

口の領域にもっとボーンが欲しいですね。Houdini のプロシージャルネットワークなら、前に戻ってジョイントを追加できます。また、Biharmonic（重調和）キャプチャを含む他のすべてのノードは、変更を反映して更新されます。クリーチャのリグの最初のセットアップを柔軟に行うことができます。

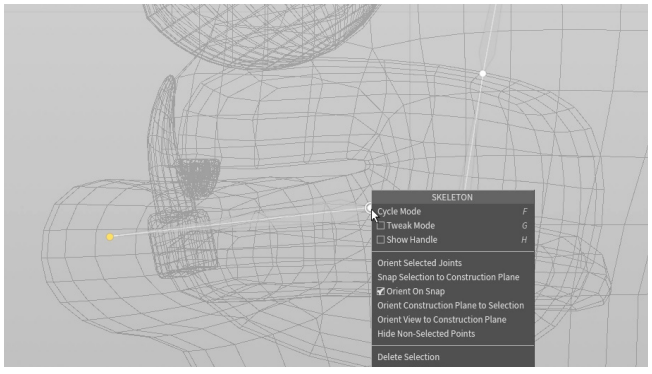


01 ネットワークビューで、skeleton ノードに **Display フラグ**を設定し、元の ***File*** ノードに **Template フラグ**を設定します。Scene View で**スペースバー + B** を 2 回押して、Right ビューに移動します。

skeleton ノードに ***Display フラグ***を設定します。***skeleton*** ノードを選択して、**Handle** ツールをオンにします。**Mode** を **Create** に設定します。スケルトンにさらにジョイントを追加できるようになります。KineFX ならこの時点でボーンを追加でき、ネットワーク内のその他のノードがプロシージャルなおかげで、後からの変更も可能です。



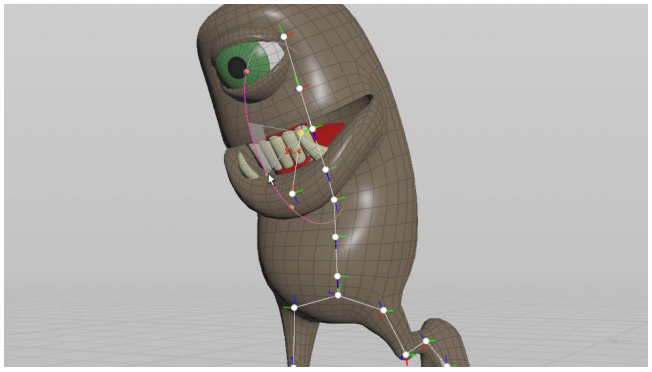
02 ***neck1*** ジョイントをクリックして描画を開始したら、顎および下唇をクリックして、2 つのジョイントを作成します。ジョイントを配置できたら、**MMB クリック**して描画を終了し、**Mode** を **Modify** に戻します。ジョイントをクリックして名前を jaw と ***lower_mouth*** に変更します。**Rig Tree** ビューで名前を変更することもできます。



03 ジョイントの位置は **Modify** モードで編集します。上部のバーで **Tweak Mode** を**オン**にすると、ジョイントをクリック＆ドラッグで移動できます。1 つのジョイントを移動すると、子のジョイントもすべて移動するため、子を元の位置に戻す作業が発生します。この動作を回避するには、**Child Compensate** をオンにします。

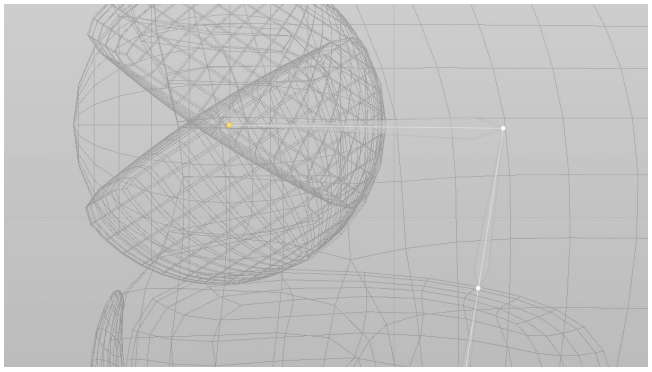
ジョイントで **RMB クリック**すると、ジョイントの分割、親子の解除、コピー、ペーストなどのオプションが表示されます。このスケルトンでは使いませんが、こうしたオプションは知っておくと便利です。

ジョイントをミラー化することもできますが、このネットワークでは別のノードを使用します。



04 ***bonedeform*** ノードに **Display フラグ**を設定し、***File*** ノードの **Template フラグ**をオフにします。ジオメトリが再構成され、新しいボーンからキャプチャが実行されます。***rigpose*** ノードを選択して、**Handle** ツールを選択します。

新しい ***jaw*** ジョイントをクリックし、下に回転させます。回転はしますが、上唇と下唇が一緒に動きます。下唇のみが影響を受ける方が良いですね。チュートリアルの後のセクションでキャプチャウェイトをペイントし、これを修正します。

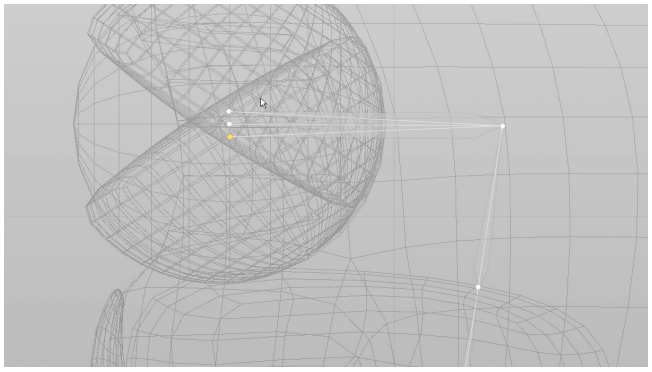


05

skeleton ノードに **Display フラグ**を再度設定し、元の ***File*** ノードに **Template フラグ**を設定します。

skeleton ノードを選択して、**Handle** ツールをオンにします。**Mode** を **Modify** に設定します。**head** ジョイントをクリックし、**Tweak Mode** を使用して、眼球の高さと揃うまで下げます。

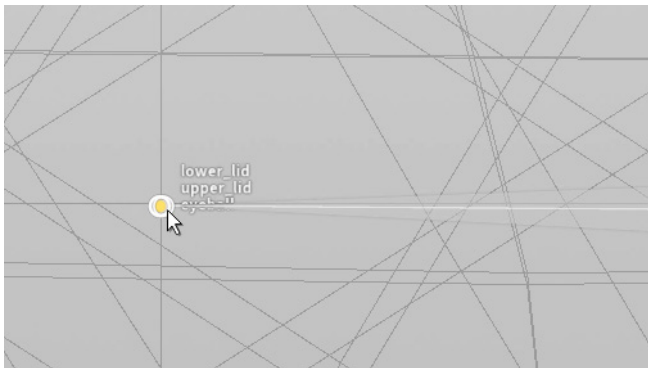
Mode を **Create** に設定します。***head*** ジョイントをクリックしたら、眼球の中央でクリックして新しいジョイントを作成します。**Modify** モードに切り替えて、**Name** でこのジョイントに ***eyeball*** と名前を付けます。



06

Mode を **Create** に設定します。***head*** ジョイントをクリックしたら、eyeball ジョイントの上をクリックして新しいジョイントを作成します。**MMB クリック**して選択を解除します。再度 ***head*** ジョイントを選択して、***eyeball*** ジョイントの下をクリックして新しいジョイントを作成します。

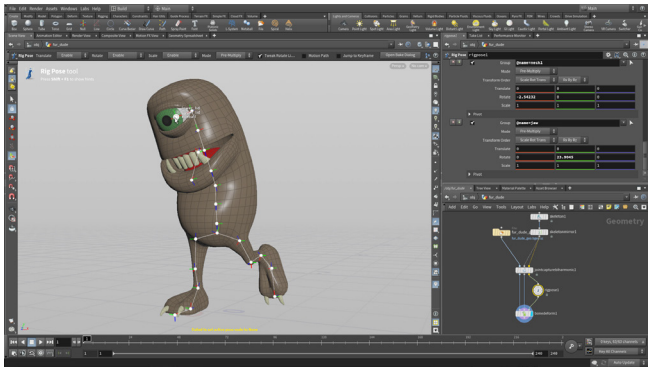
Modify モードに切り替えて、**Name** でジョイントにそれぞれ ***upper_lid*** と ***lower_lid*** と名前を付けます。アニメーション時にはこれらのジョイントでまぶたを回転させますが、ジョイントは目の中央に配置しておく必要があります。



07

ジオメトリの Template フラグをオフにします。***upper_lid*** ジョイントを選択し、**Tweak Mode** を使用して ***eyeball*** ジョイントに重なるようにドラッグします。***lower_lid*** でもこの作業を繰り返して、***eyeball*** ジョイントに重ねます。

これで、3 つすべてのジョイントが同じ場所に揃いました。後で、それぞれ個別にジオメトリをアタッチして、独立してアニメートできるようにします。



08

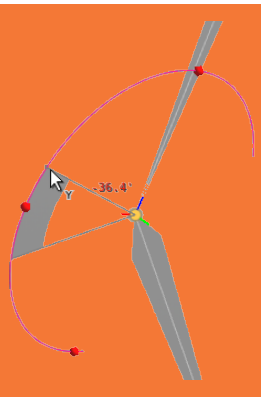
bonedeform ノードに **Display フラグ**を設定します。ジオメトリが再構成され、新しいボーンを使用してキャプチャされます。

ジオメトリはまだ適切にアタッチされていないので、新しいジョイントにポーズを付けることはできません。いくつかの手順を行い、体と舌にキャプチャウェイトをペイントした後で、ようやく可能になります。

ジョイントの向き

ジョイントベースのシステムでは、各ジョイントはポイントです。ジョイントの向きが非常に重要なのは、それによって、フォワードキネマティクスでジョイントがどう回転するかが決まるからです。

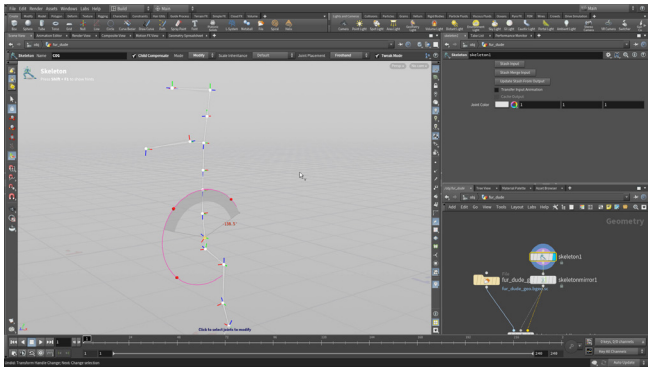
KineFX に含まれるツールを使用すると、チェーンに沿って次のジョイントの方を向くようにすることができます。また、**Child Compensate** オプションを使って手動でジョイントを回転すると、チェーン内の他のジョイントに影響を与えずに、ジョイントの向きを変更できます。



パート 4

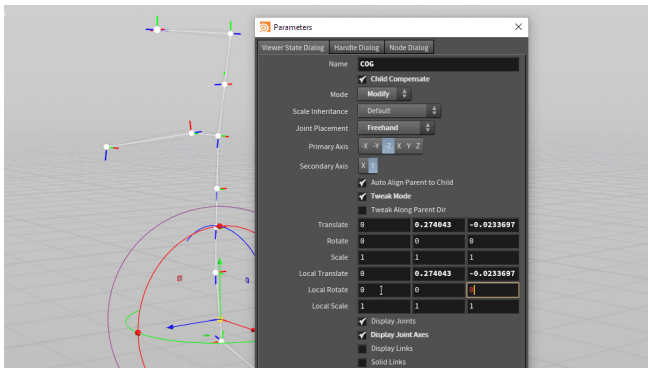
ジョイントの向き

キャラクタをアニメートするときは、ジョイントの向きがリグの操作に大きく影響します。このパートでは、一部のジョイントの向きを手動で変更します。その後、Orient Joints ノードを使用して、残りすべてのジョイントの向きをマイナス Z 軸基準に変更します。後の工程でリグを評価する際、必要に応じてここに戻って向きを微調整してください。



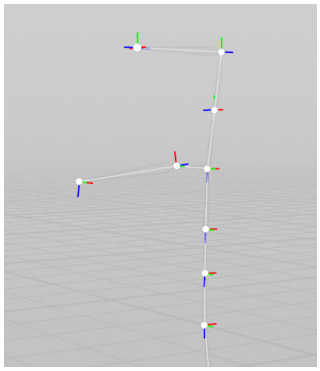
01 もう一度、***skeleton*** ノードに **Display フラグ**を設定します。Scene View で **RMB クリック**して **Display Joint Axes** を選択し、ジョイントの向きが確認できるようにします。***COG*** ジョイントを選択して **RMB クリック**し、**Show Handle** を選択します。これで、ジョイントを回転させると、リグ全体が一緒に動きます。**Ctrl + Z** を押して元に戻しましょう。

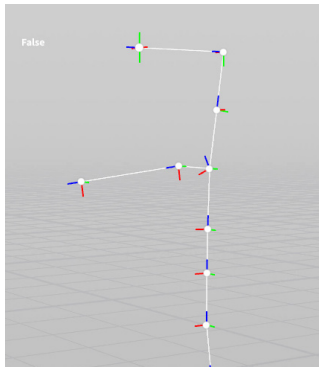
Child Compensate チェックボックスをオンにすると、残りのスケルトンに影響を与えることなく、ジョイントを回転できます。**Ctrl** キーを使用して、45 度のインクリメントに拘束します。



02 Scene View で、**P を押して**このジョイントのパラメータを表示します。ワールドで COG の向きを変更するために、**Rotate** と **Local Rotate** を **0, 0, 0** に設定します。

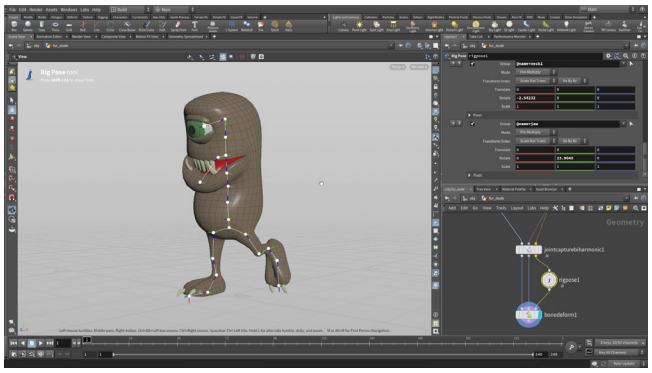
何もない空間でクリックして COG ジョイントの選択を解除したら、***pelvis*** ジョイントをクリックします。**Rotate** と **Local Rotate** を **0, 0, 0** に設定します。





03 Skeleton ノードの後に、**Orient Joints** ノードを挿入します。このノードは、デフォルトではプラス Z 軸を基準にジョイントの向きを変更します。**Orient Group** の横の矢印をクリックして、Scene View ですべてのジョイントを選択します。**Ctrl を押し**ながら ***COG*** ジョイント、***pelvis*** ジョイント、***neck1*** ジョイントを選択し、選択から除外します。**Enter** を押します。

これで、選択解除した 3 つのジョイント以外、すべてのジョイントがプラス Z 軸を基準とした向きになります。



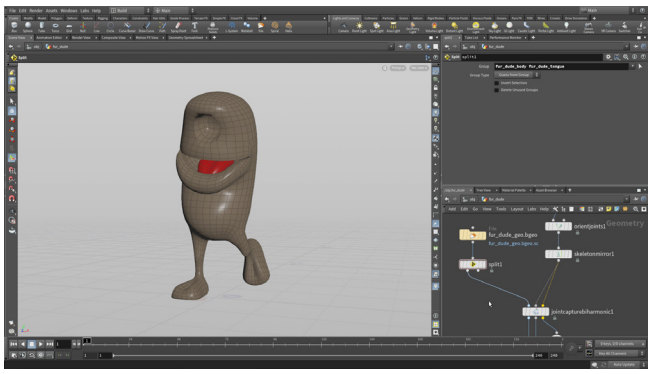
04 **Bone Deform** ノードに Display フラグを再度設定します。他のすべてのノードが更新され、新しいジョイントの向きが受け入れられます。

ジョイントの向きを変える前後では、違いはないように見えますが、アニメーション時のキャラクタのポーズ決めや操作に影響してきます。

パート 5

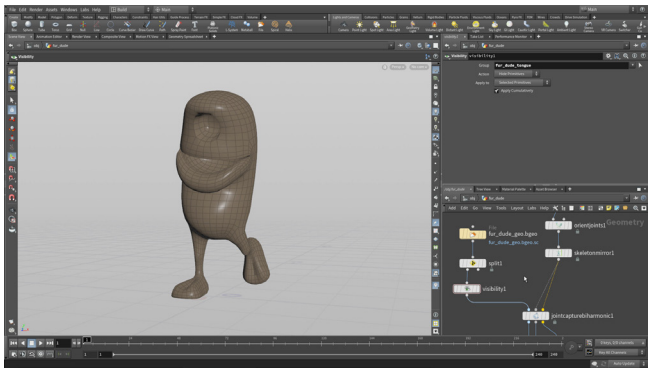
キャプチャジオメトリの取り付け

現在はスケルトンジョイントを使用してジオメトリをキャプチャし、キャラクタの各ポイントにキャプチャウェイトを割り当てています。これをさらに詳細にコントロールするには、そのジョイントの影響を延長するジョイントにカーブを取り付けます。このようにすると、かなり素早く素早く目的を達成することができます。



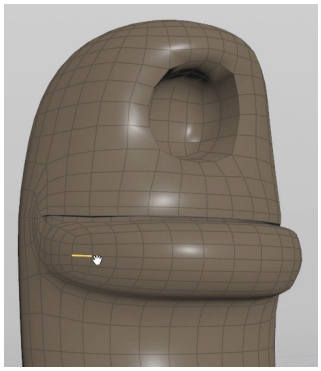
01 ネットワークビューで、**Tab > Split** を選択して、***File*** ノードと ***jointcapturebiharmonic*** ノードの間にノードを配置します。split ノードの 1 つ目の出力を、***jointcapturebiharmonic*** ノードの 1 つ目の入力に接続します。

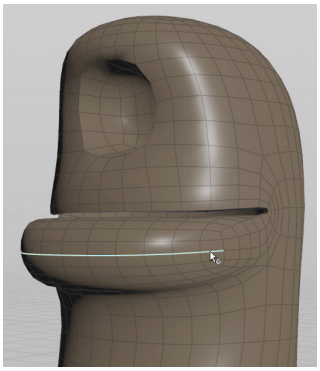
Group フィールドのプルダウンメニューをクリックして、***fur_dude_body*** グループと ***fur_dude_tongue*** グループを選択します。これで、この 2 つは split ノードの 1 番目の出力に送られ、眼球、歯、つめなど残りのパーツは 2 番目の出力に送られるようになります。このジオメトリにウェイトをペイントしてから、残りのパーツを別の方法を使ってスケルトンにバインドします。



02 ネットワークビューで、**Tab > Visibility** を選択して、***Split*** ノードと ***jointcapturebiharmonic*** ノードの間にノードを配置します。**Group** フィールドのプルダウンメニューをクリックして、***fur_dude_tongue*** グループを選択します。

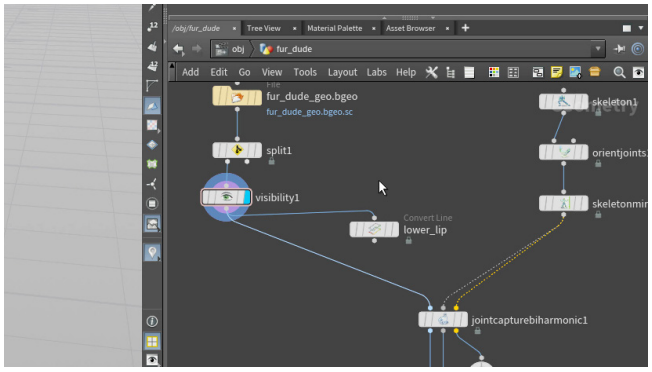
このノードは選択したジオメトリを非表示にします（削除はしません）。つまり、Paint Capture Weights ツールを使用する際に重要となるポイント番号とプリミティブ番号が変更されません。この処理でジオメトリを非表示にするたびに、これらの情報が変更されてしまっては大変です。





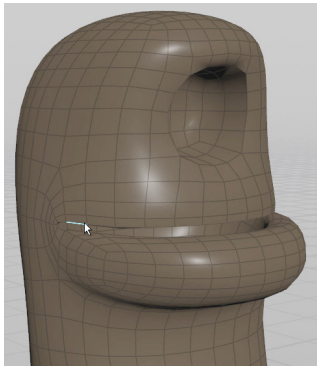
03 **visibility** ノードに **Display フラグ**を設定します。

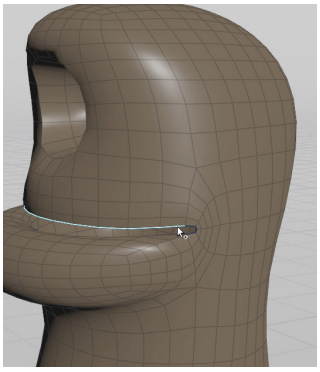
S を押して Select ツールにしたら、**3** を押してエッジ選択を有効にします。下唇の左側中央でエッジを選択したら、**Shift + A** を押して反対側のエッジを選択します。



04 **Tab > Curve from Edges** を押すと、選択されたエッジがジオメトリから抽出されます。このノードの名前を ***lower_lip*** に変更します。

curvefromedges ノードが ***File*** ノードと ***jointcapturebiharmonic*** ノードに間に配置されています。それを横に移動して、***visibility*** ノードの出力を ***jointcapturebiharmonic*** ノードの 1 つ目の入力に再接続します。これにより、***curvefromedges*** ノードが横に分岐します。



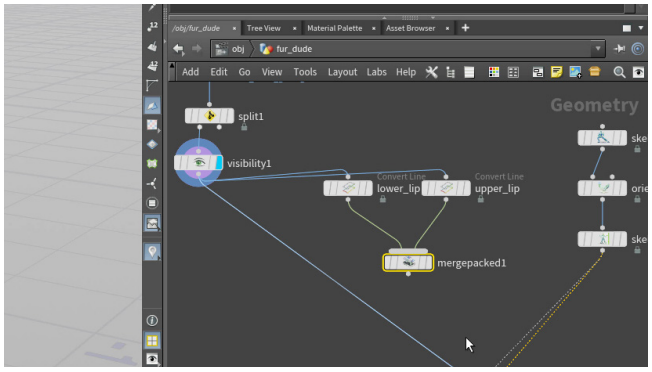


05 **visibility** ノードに **Display フラグ**が設定されていることを確認します。

S を押して Select ツールにしたら、**3** を押してエッジ選択を有効にします。上唇の左側でエッジを選択したら、**Shift + A** を押して反対側のエッジを選択します。

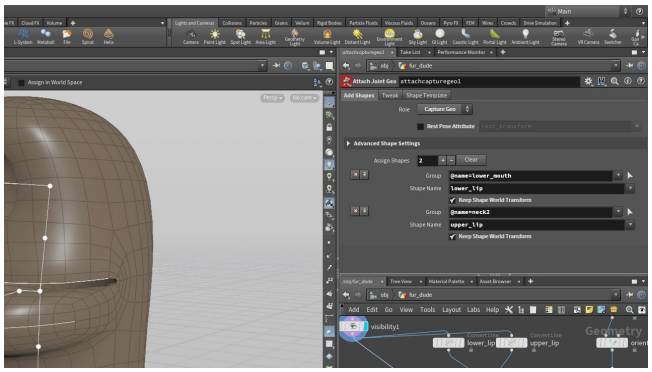
Tab > Curve from Edges を押すと、選択されたエッジがジオメトリから抽出されます。このノードの名前を ***upper_lip*** に変更します。

このノードを、もう片方のノードで行ったのと同じように分岐させます。



06 **Tab > Merge Packed** を選択し、このノードを抽出した 2 つのカーブのノードの下に配置します。その後、その 2 つのノードの出力を ***mergepacked*** ノードに接続します。

Tab > Attach Capture Geo を押します。このノードをネットワークに追加したら、***skeletonmirror*** ノードを 1 つ目の入力に接続し、***mergepacked*** ノードを 2 つ目の入力に接続します。



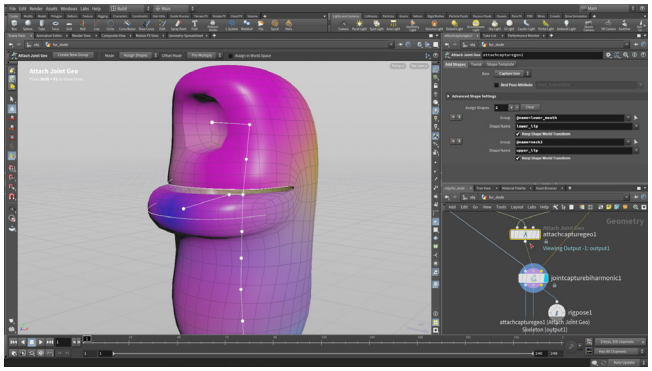
07 ***add_capture_geometry*** ノードの出力を、***jointcapturebiharmonic*** ノードの中央の入力に接続します。**Advanced Shape Settings** で、**Assign Shapes** の +（プラス）記号ボタンを 2 回クリックします。1 つ目は次のように設定します。

- **Group** を ***@name=lower_mouth*** にする
- **Shape Name** を ***lower_lip*** にする

2 つ目は次のように設定します。

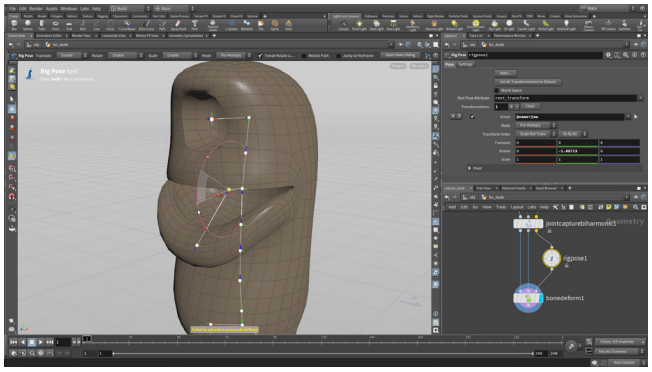
- **Group** を ***@name=neck2*** にする
- **Shape Name** を ***upper_lip*** にする

どちらの形状でも、**Keep Shape World Transform** を**オン**にします。



08 ***jointcapturebiharmonic*** に **Display フラグ**を設定します。ここで ***add_capture_geometry*** ノードを**バイパス**して、キャプチャウェイトの違いを表示します。

このノードがオンのとき、取り付けられたカーブは、関連付けられたジョイントへのジオメトリのキャプチャを補助します。これにより、キャラクタのセットアップをより細かくコントロールできるようになります。



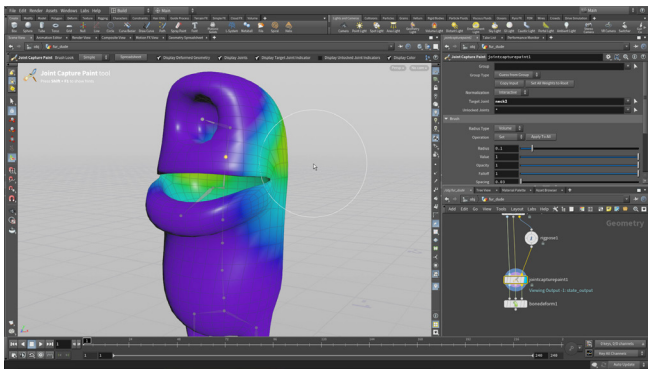
09 ***bonedeform*** ノードに **Display フラグ**を設定します。***rigpose*** ノードをクリックしたら、パラメータエディタで **Transformations** の横の **Clear** ボタンをクリックします。これでジョイントがリセットされます。**Handle** ツールをアクティブにして、Scene View で ***jaw*** ジョイントをクリックします。それを下に**回転**させて唇を下げます。

この操作で、腹部のパーツがまだ変形していることが分かります。これを修正するには、キャプチャウェイトをペイントして、キャプチャウェイトを各種ジョイントに割り当て直す必要があります。

パート 6

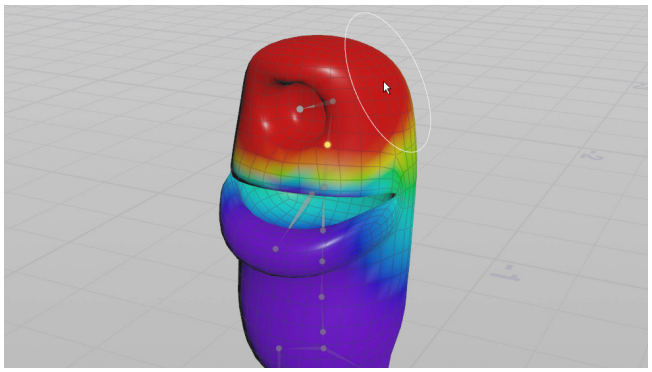
キャプチャウェイトのペイント

Biharmonic（重調和）キャプチャは、スケルトンの各種ボーンに関連付けられたキャラクタのジオメトリに、ウェイトを追加します。次は、新しいノードを追加して、ブラシワークフローによってキャプチャウェイトを調整します。このクリーチャのレッスンで目指すのは、上唇が下唇のジョイントの影響を受けないようにすることと、足の領域のウェイトを微調整することです。



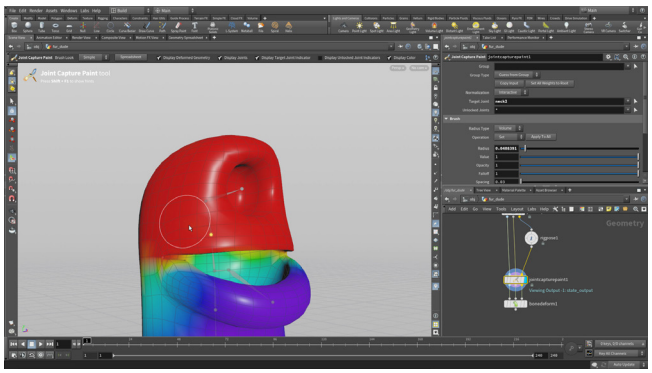
01 ネットワークビューで、**Tab > Joint Capture Paint** を押して、ノードを ***bonedeform*** の真上に配置し、3 つすべてのコネクタを接続します。**Display フラグ**を設定してから、**Target Joint** の横のプルダウンメニューをクリックして、***neck2*** ジョイントを選択します。

Scene View には、カーソルに大きい円形のペイントアイコンが表示され、これを使ってウェイトをペイントします。頭部の領域をペイントすると、ジオメトリのこの部分が ***neck2*** ジョイントにキャプチャされます。



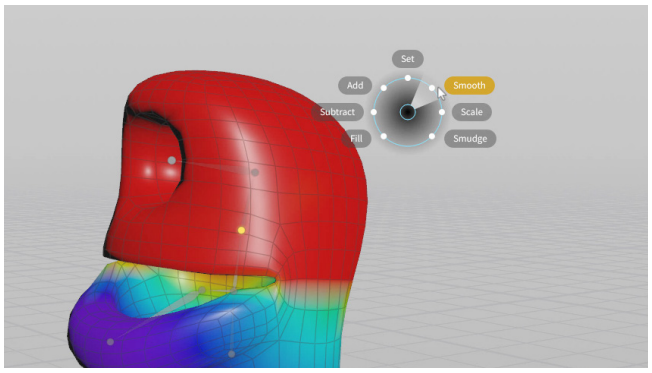
02 頭部と上唇の領域を neck2 ジョイントにキャプチャする必要があります。影響の度合いが最も強い領域のジオメトリは、赤くなります。

頭の上部では太いストロークを使用します。その後、マウスの**スクロールホイール**を使用してブラシの半径を小さくするか、パラメータエディタの **Brush** タブに移動して半径を変更します。



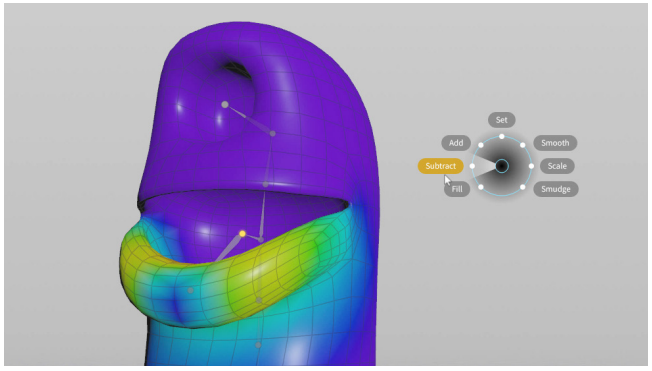
03 上唇をペイントして、この領域に対する ***neck2*** ジョイントの影響を強くします。ペイントするのは上唇のみで、下唇には及ばないようにします。間違えたら、**Ctrl + Z** でストロークを取り消しましょう。タンブルして口の中が見えるようにして、上部もペイントします。

オペレーションコントロールツールバーには、**Display Deformed Geometry**、**Display Joints**、**Display Color** のオプションが表示されています。これらのオン/オフを切り替えることで、キャプチャウェイトを評価しながらペイントできます。



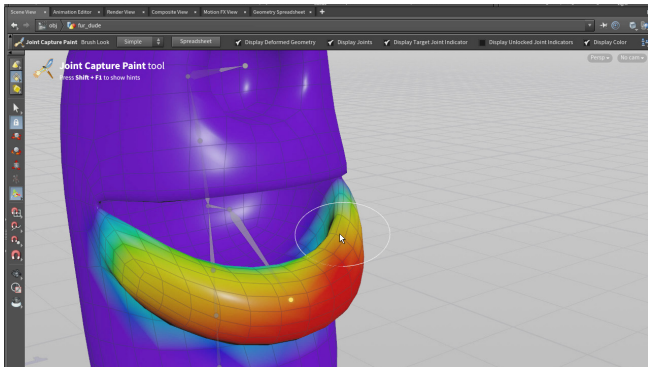
04 **F を押して Smooth** を選択し、この部分のペイントを周りの領域のキャプチャウェイトに馴染ませます。ブラシの半径を少し大きくして、滑らかにしましょう。

完了したら、***rigpose*** ノードをクリックします。**Handle** ツールをアクティブにして、Scene View で ***jaw*** ジョイントをクリックします。それを下に回転させて唇を下げます。今回は、上唇は動かず、下唇およびその下のパーツが回転しているのが分かります。



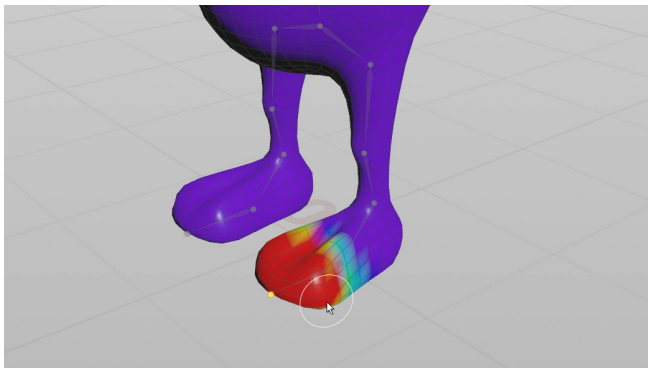
05 ***jointcapturepaint*** ノードを選択します。**Capture Region** フィールドのプルダウンメニューをクリックして、***jaw*** ジョイントを選択します。**F を押して Subtract** を選択し、頭頂や目の領域に対する jaw ジョイントの影響をなくします。

この方法を使用すると、腹部に対する jaw ジョイントと lower_mouth ジョイントの影響もなくすことができます。そうすると、口が動くとき、腹部があまり影響を受けなくなります。

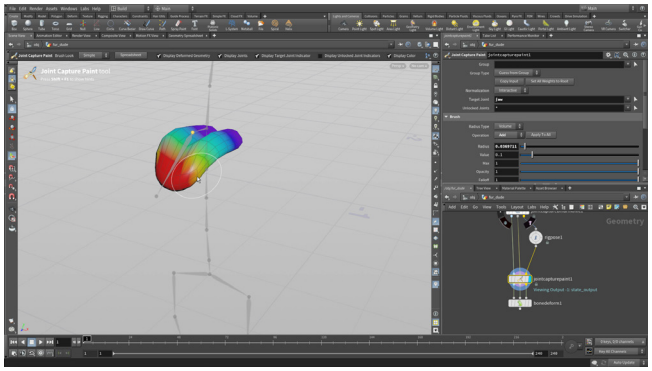


06 時々前に戻って、jaw をテストしながら、調整します。余分な領域に jaw が影響しなくなったら、完了です。再度スムースをかけて、馴染ませてもよいでしょう。

次は、スケルトンの別のジョイントにウェイトをペイントします。Biharmonic（重調和）のおかげで、脚や足に問題はないはずですが、***rigpose*** を使用してリグをテストし、改善点があればウェイトをペイントして微調整してください。



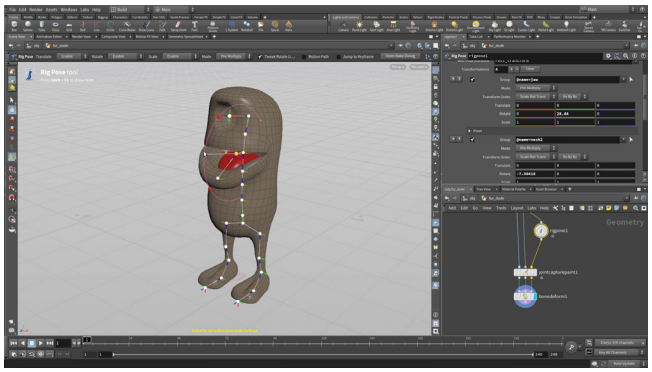
07 toe ジョイントを選択して、足の先端が左右の toe ジョイントにキャプチャされていることを確認します。後ほど、リバースフットのセットアップを構築して、ジオメトリをつま先にアタッチします。



08 visibility ノードで **Apply to** を **Non-Selected Primitives** に設定します。すると、舌だけが表示されます。

デフォルトで適切にウェイトが付いているはずですが、必要に応じてウェイトをペイントできます。***jointcapturepaint*** ノードを選択して、***neck1*** ジョイントと ***jaw*** ジョイントのウェイトをペイントします。

完了したら、***visibility*** ノードを再度**バイパス**します。



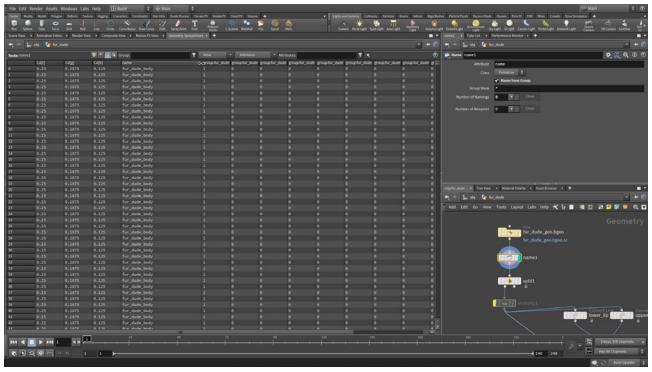
09 visibility ノードに **Bypass フラグ**を設定します。

完了したら、***rigpose*** ノードをクリックします。**Handle** ツールをアクティブにして、Scene View で ***jaw*** ジョイントをクリックします。それを下に回転させて唇を下げます。すべてのパーツの動きに満足したら、リジッドジオメトリをキャプチャする準備の完了です。

パート 7

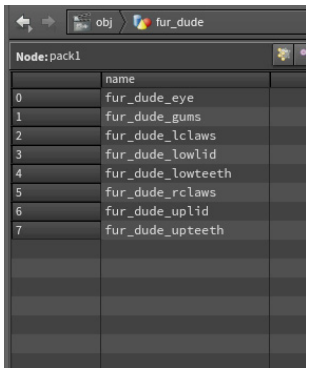
リジッドジオメトリのキャプチャ

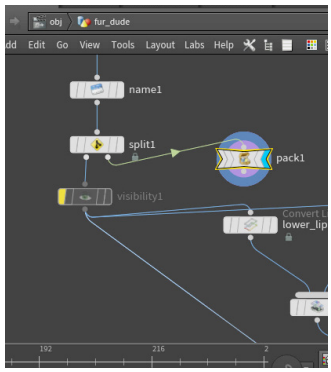
パート 5 で、目、歯、つめのジオメトリを分割しました。ここでは、このジオメトリをパック化し、その後 Capture Packed Geometry ノードを使用して、各パーツをジョイントに割り当てます。KineFX では、親子化はジオメトリレベルでは使用できません。この操作は、各オブジェクトをスケルトンの子にすることに相当します。



01 ネットワークビューで、**Tab > Name from Groups** を選択して、ノードを ***File*** ノードと ***split*** ノードの間に配置します。**Group Mask** を * に変更します。これで、すべてのグループが name アトリビュートになります。

これを表示するには、**Geometry Spreadsheet** タブをクリックして、左上の **Primitives** ボタンをクリックします。下にスクロールすると、name アトリビュートがあることと、グループ名がすべてのジオメトリの値として使用されていることを確認できます。

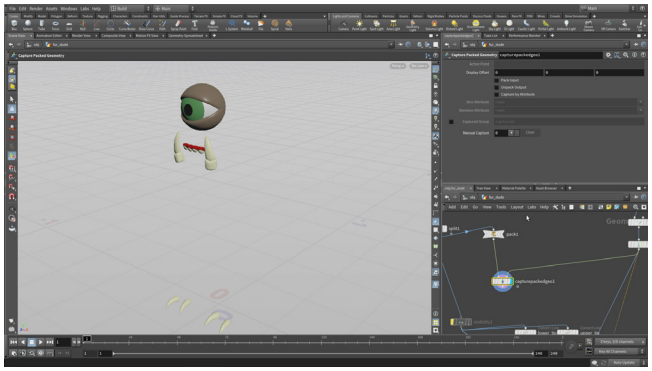




02 ネットワークビューで **Tab > Pack** を選択し、***split*** ノードの右側にノードを配置します。***split*** ノードの 2 番目の出力を ***pack*** ノードに接続して、**Display フラグ**を設定します。すると、Scene View に目、歯、つめのパーツが表示されます。

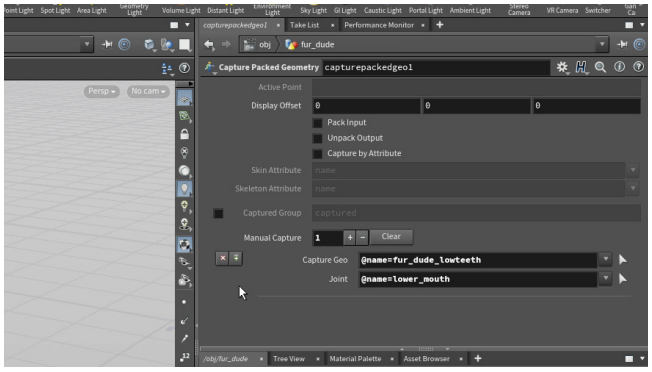
Path Attribute チェックボックスを**オフ**にして、**Name Attribute** チェックボックスを**オン**にします。**Transfer Attributes** を **name** に設定します。

ネットワークビューの右上の矢印を使用して、**Split** を選択します。**Geometry Spreadsheet** に、8 つのパックプリミティブが表示されます。これらは、リジッドジオメトリとしてスケルトンにキャプチャできます。



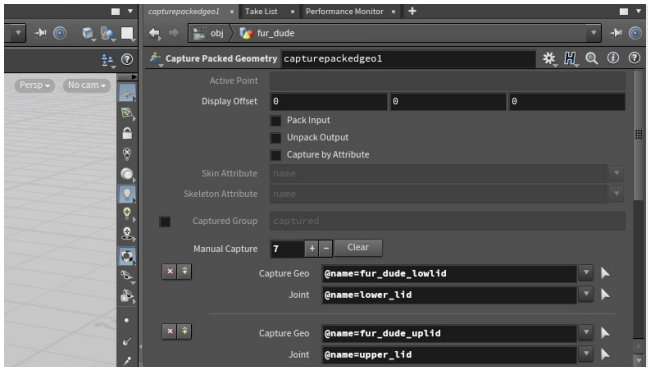
03 ネットワークビューで **Tab > Capture Packed Geometry** を選択して、そのノードを ***pack*** ノードの下に配置します。pack ノードの出力を、***capturepackedgeo*** ノードの 1 つ目の入力に接続します。次に、skeletonmirror ノードの出力を、***capturepackedgeo*** ノードの 2 つ目の入力に接続して、***capturepackedgeo*** ノードに Display フラグを設定します。

これだけではまだ何も起こりません。ボーンとキャプチャしたいジオメトリを関連付ける必要があります。



04 パラメータエディタで、**Manual Capture** の横にある **+（プラス）記号**をクリックします。**Capture Geo** の横の**矢印**をクリックして、Scene View で***下の歯***を選択します。ワンクリックで済むのは、下の歯はパックグループの 1 つだからです。**Enter** を押して確定します。

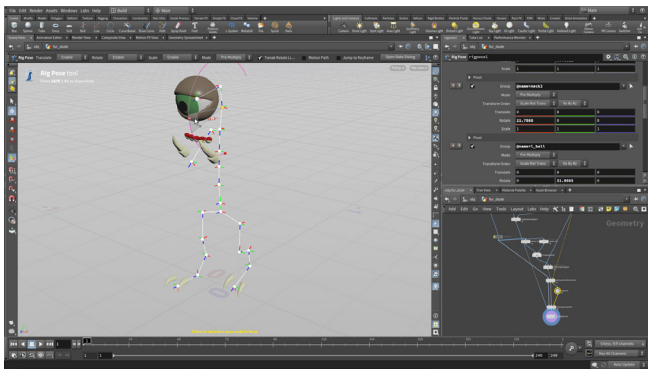
次に、Joint の横の**矢印**をクリックします。ジオメトリが消え、スケルトンを表すラインとポイントが表示されます。***lower_mouth*** ジョイントを選択したら、カーソルを Scene View 上に置いたまま **Enter** を押して確定します。



05 +(プラス)記号をクリックして、**Capture Geo** の横の**矢印**をクリックします。Scene View で、Shift を押しながら ***fur_dude_upteeth*** と ***fur_dude_gums*** を選択します。**Enter** を押して確定します。次に、**Joint** の横の**矢印**をクリックします。***neck2*** ジョイントを選択して、**Enter** を押します。

この手順をさらに 2 回繰り返して、以下を関連付けます。

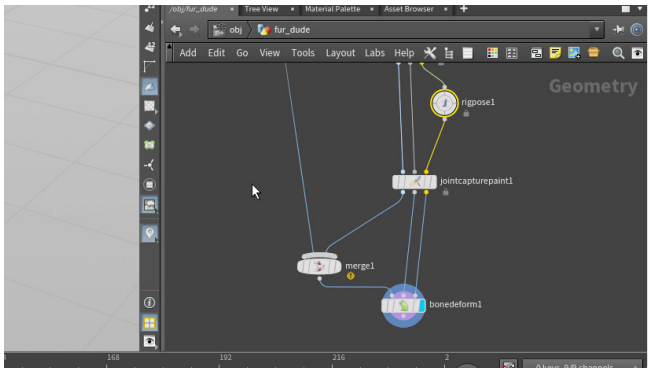
- ***fur_dude_rclaws*** と ***r_toe***
- ***fur_dude_lclaws*** と ***l_toe***
- ***fur_dude_eye*** と ***eyeball***
- ***fur_dude_uplid*** と ***upper_lid***
- ***fur_dude_lowlid*** と ***lower_lidl***



06 ***capturepackedgeo*** ノードの出力を、***bonedeform* ノード**の**1 つ目の入力**に接続します。***bonedeform*** ノードに Display フラグを設定します。

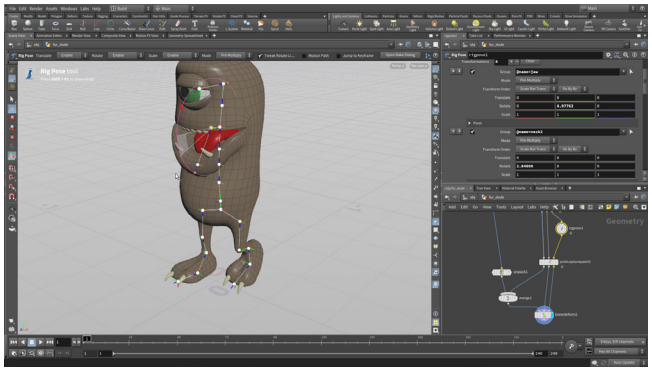
rigpose ノードを選択したら、Scene View でジョイントを選択して回転します。ジオメトリがジョイントにキャプチャされ、変形がないことを確認できます。

目とまぶたのジョイントは重なっているため、ジョイントをクリックしたら、小さいポップアップメニューを使用していずれかを選択します。必要なジョイントにたどり着くまで、数回クリックする必要がある場合もあります。選択したジョイントを使用して、目とまぶたを別々に回転します。



07 ネットワークビューで **Tab > Merge** を選択して、ノードを ***capturepackedgeo*** ノードと ***bonedeform*** ノードの間に配置します。***capturelayerpaint*** ノードを merge ノードに接続します。***merge*** ノードで、***capturelayerpaint*** の横の青い上向き矢印を押して、入力の順番を変更します。

これですべてが変形するようになりました。しかし、パックジオメトリには色が付いていて、体と舌はグレーです。merge ノードにも問題があります。一方には Primitive アトリビュートとしてカラー(Cd)が設定されているのに対し、もう一方には Point アトリビュートとしてカラーが設定されています。



08 ネットワークビューで **Tab > Unpack** を選択し、***capturepackedgeo*** ノードと merge ノードの間にノードを配置します。**Iterations** を **2**、**Transfer Attributes** を * **^Cd** に設定します。* ですべてのキャプチャアトリビュートが取り込まれます。また **^Cd** により、元のカラーアトリビュートが除去されません。これで、どちらもポイントカラーを使用するようになり、リグが正しく表示されるようになりました。

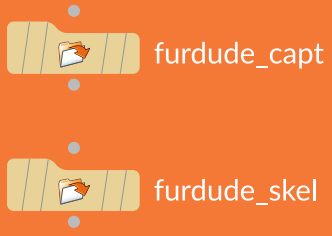
rigpose ノードを選択し、Scene View でジョイントを選択して回転します。すべてのキャプチャされたジオメトリが一緒に動くのを確認できます。

作業内容を**保存**します。

ネットワークの統合

ジオメトリのキャプチャ、ウェイトのペイント、アニメーション用のジオメトリの準備のために作成したネットワークは適切に動作しますが、変更を加えると更新に時間がかかります。

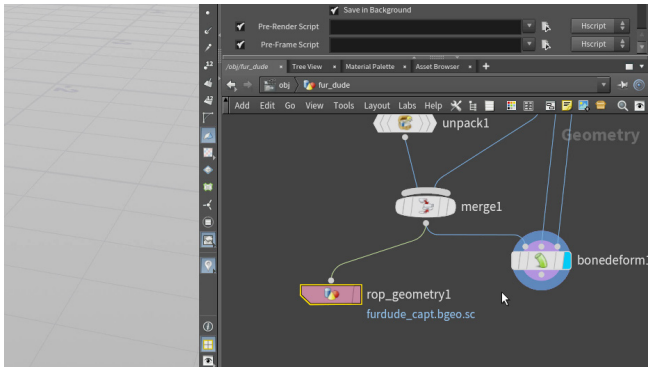
より効率的なリグを作成するには、ネットワークを統合し、キャプチャウェイトをジオメトリに格納した単一ファイルにします。このファイルを Bone Deform SOP に接続すると、最初にジオメトリをキャプチャするのに使用したボーンと同じボーンのスケルトンであれば、効率的に変形されるようになります。



パート 8

キャプチャリグのデジタルアセットの作成

キャプチャしたジオメトリとスケルトンを、デジタルアセットにラップして、アニメーションリグの土台として使用します。まず、キャプチャウェイトやスケルトンとともにジオメトリをエクスポートして、デジタルアセットファイルに埋め込みます。こうすると、キャラクタのアニメートで効率の良いリグが得られます。

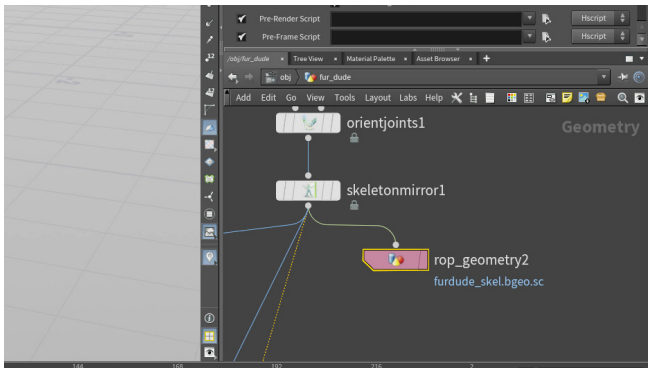


01

ネットワークビューで **Tab > ROP Geometry Output** を選択し、***merge*** ノードの下にノードを配置します。***merge*** ノードの出力を ***rop_geometry*** ノードの入力に接続します。パラメータエディタで、**Output File** を次のように設定します。

$HIP/geo/furdude_capt.bgeo.sc

Save to Disk をクリックして、ジオメトリを ***geo*** ディレクトリに保存します。このジオメトリは、レッスンの最初にインポートしたジオメトリと同じように見えますが、変形を可能にするキャプチャアトリビュートなどの重要な情報が含まれています。

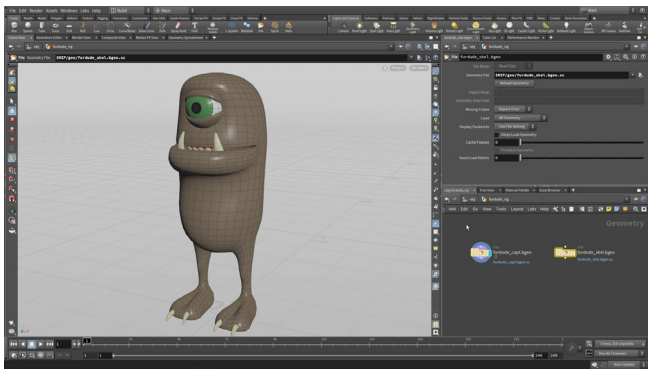


02

ネットワークビューで **Tab > ROP Geometry Output** を選択し、***skeletonmirror*** ノードの横にノードを配置します。***skeletonmirror*** ノードの出力を ***rop_geometry*** ノードの入力に接続します。パラメータエディタで、**Output File** を次のように設定します。

$HIP/geo/furdude_skel.bgeo.sc

Save to Disk をクリックして、ジオメトリを ***geo*** ディレクトリに保存します。このジオメトリはスケルトンを表し、アニメーションリグアセットの構築に使用できます。

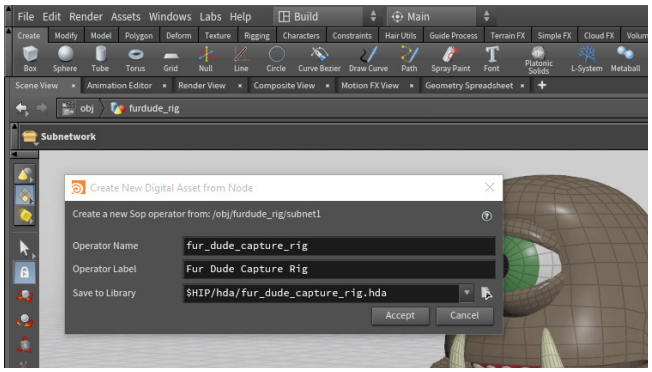


03

オブジェクトレベルに戻ります。オブジェクトの名前を ***fur_dude_capture*** に変更します。その **Display フラグ**をオフにします。

Create シェルフの **File** ボタンをクリックします。***$HIP*** をクリックして、***geo*** ディレクトリに移動し、***furdude_capt.bgeo.sc*** ファイルを選択します。**Enter** を押して原点に配置したら、新しいオブジェクトノードの名前を ***furdude_rig*** に**変更**します。**ダブルクリック**してこのオブジェクトの中に入ります。***File*** ノードを **Alt ドラッグ** して、2 つ目のノードを作成します。**Geometry File** を ***$HIP/geo/furdude_skel.bgeo.sc*** に変更します。

それを ***furdude_skel.bgeo*** という**名前に変更**し、***furdude_capt.bgeo*** ノードに **Display フラグ**を設定します。

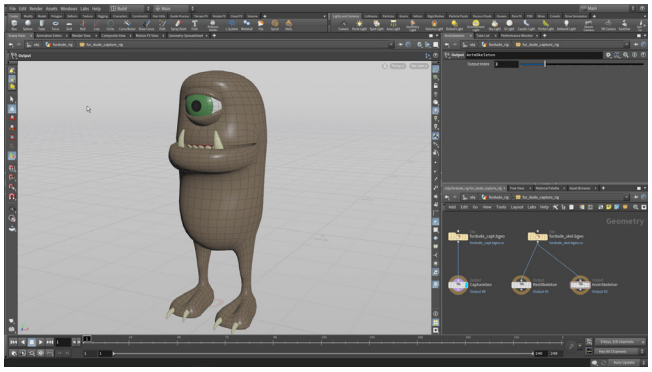


04

2 つの File ノードを選択し、**Assets** メニューから **New Digital Asset from Selection** を選択します。次のように設定します。

- **Operator Name** を ***fur_dude_capture_rig*** にする
- **Operator Label** を ***Fur Dude Capture Rig*** にする
- **Save to Library** を ***$HIP/hda/fur_dude.hda*** にする

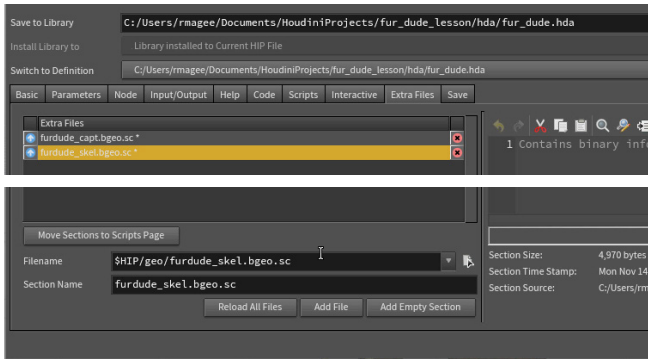
Accept をクリックして、HDA ファイルを作成します。Edit Operator Type Properties ウィンドウが表示されます。**Maximum Outputs** を **3** に設定し、**Accept** をクリックします。サブネットを ***fur_dude_capture_rig*** という**名前に変更**します。



05 このノードを**ダブルクリック**して、サブネットの中に入ります。**Tab > Output** を選択して、output ノードを ***fur_dude_geo_capt File*** ノードの下に配置します。***File*** ノードを ***output*** ノードに接続します。名前を ***CaptureGeo*** に変更します。

Alt ドラッグ操作を 2 回行い、新しい Output ノードを 2 つ作成します。2 つ目の名前を ***RestSkeleton*** に変更し、**Output Index** を **1** に設定します。3 つ目の名前を ***AnimSkeleton*** にして、**Output Index** を **2** に設定します。***fur_dude_skel.bgeo File*** ノードを、2 つ目および 3 つ目の ***output*** ノードに接続します。

CaptureGeo output ノードに **Display フラグ**を設定します。

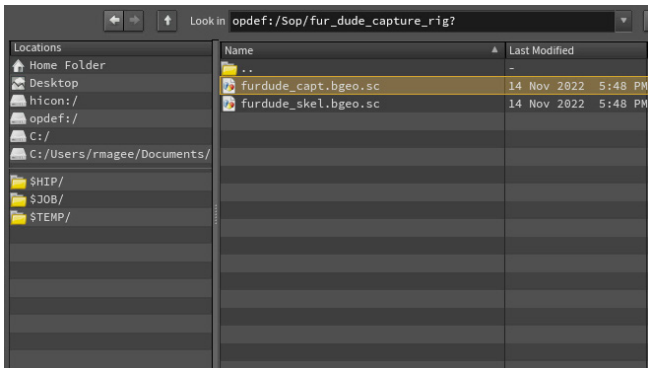


06 **Asset** メニューから、**Edit Asset Properties > Fur Dude Capture Rig** を選択します。Edit Operator Type Properties ウィンドウが開きます。**Extra Files** タブをクリックします。

左下の **Filename** の横にある選択ボタンをクリックして、***$HIP/geo/fur_dude_capt.bgeo.sc*** に移動します。**Accept** をクリックします。**Add File** をクリックします。

この手順を ***fur_dude_skel.bgeo.sc*** ファイルでも繰り返します。これで、これらのファイルがデジタルアセットファイル内部に配置できました。完全なパッケージとして他の人と共有しやすくなります。

Accept をクリックして完了します。

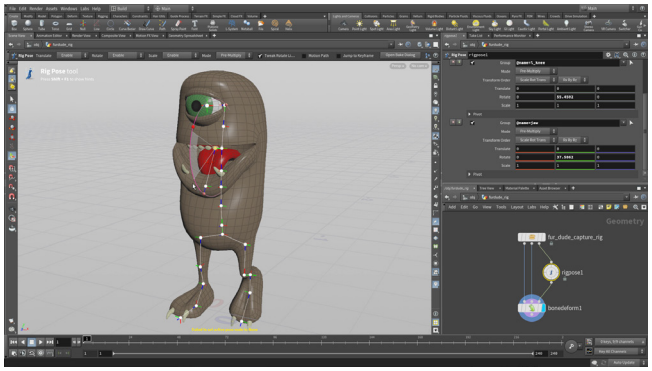


07 最初の file ノードで、**Geometry File** の横にある**選択**アイコンをクリックします。**Locations** サイドバーで ***opdef:/*** を選択し、***Sop*** ディレクトリをダブルクリックしたら、***fur_dude_capture_rig*** フォルダをクリックします。***fur_dude_capt.bgeo.sc*** ファイルを選択し、**Accept** を押します。これにより、以下の opdef エクスプレッションが作成されます。

```
opdef:/Sop/fur_dude_capture_rig?furdude_capt.bgeo.sc
```

この手順を ***fur_dude_skel.bgeo.sc File*** ノードでも繰り返します。

Assets メニューから **Lock Asset > Fur Dude Capture Rig** を選択します。**Save Changes** をクリックして、このアセットのコンテンツを保護します。これらのファイルは、必要に応じて後からロックを解除して、更新できます。



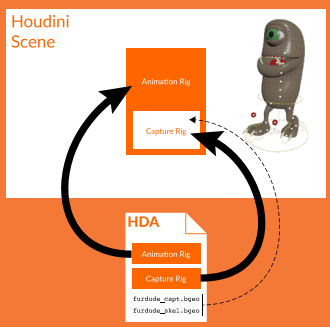
08 1 つ上のレベルに移動すると、3 つの出力があるキャプチャリグを確認できます。**Tab > Bone Deform** を選択し、このノードを下に配置します。***fur_dude_capt_rig*** の 3 つの出力を、***bonedeform*** ノードの 3 つの入力に接続します。3 つ目のチェーンの中央に、rigpose ノードを追加します。***bonedeform*** ノードに **Display フラグ**を設定します。

rigpose ノードを選択して、**Handle** ツールがアクティブなことを確認します。すべてのスケルトンジョイントが再度表示されます。スケルトンに**ポーズ**を付け、変形が前と同じように動作しているかをテストします。

HDA

このようなノードのネットワークは、Houdini デジタルアセット（HDA）として保存されます。これはディスク上の 1 ファイルで、簡単に共有できます。キャプチャリグはディスクから参照されるアセットです。次のセクションで、内部にキャプチャリグをネストしたアニメーションリグを構築します。どちらのファイルも、bgeo ファイルとともに単一の HDA ファイル内に保存されます。

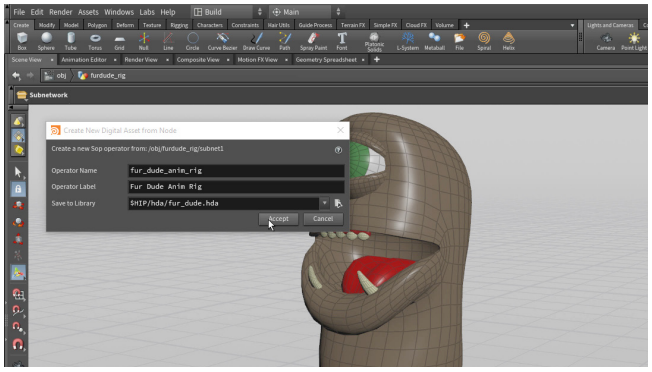
ディスク上の HDA ファイルが変更されると、アニメータがシーンファイル内で作業しているアセットのインスタンスが更新されます。



パート 9

アニメーションリグアセットの作成

このパートでは、内部にキャプチャリグをネストした 2 つ目のデジタルアセットを作成します。この新しいアセットをアニメートして、キャラクタの最終的なモーションを作成します。また、この新しいアセットに、アニメーションをサポートするインバースキネマティクスや Aim 拘束などのリギングツールを追加していきます。コントロールを追加するたびにテストしていきます。このためには、ロックされたテストリグをセットアップし、2 つ目の Scene View ペインに表示させます。



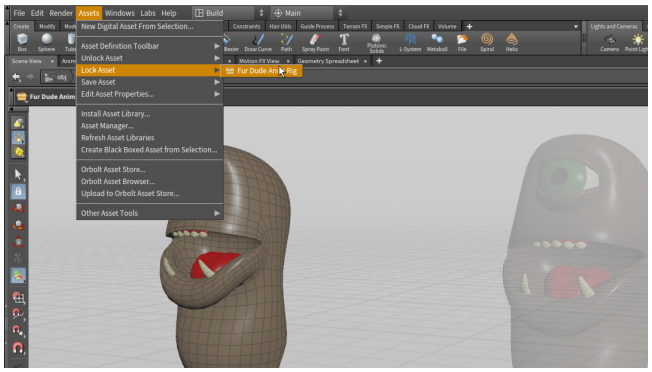
01

3 つのノードを選択し、**Assets** メニューから **New Digital Asset from Selection** を選択します。次のように設定します。

- **Operator Name** を ***fur_dude_anim_rig*** にする
- **Operator Label** を ***Fur Dude Anim Rig*** にする

Save to Library では、ブラウザボタンをクリックします。$HIP をクリックし、HDA ディレクトリをダブルクリックします。fur_dude.hda ファイルを選択し、**Accept** をクリックします。これで、***$HIP/hda/fur_dude.hda*** に設定されました。

Accept をクリックし、Edit Operator Type Properties ウィンドウで再度 **Accept** をクリックします。これで、新しいアセット定義が同じ HDA ファイルに追加されました。サブネットを ***fur_dude_anim_rig*** という**名前に変更**します。



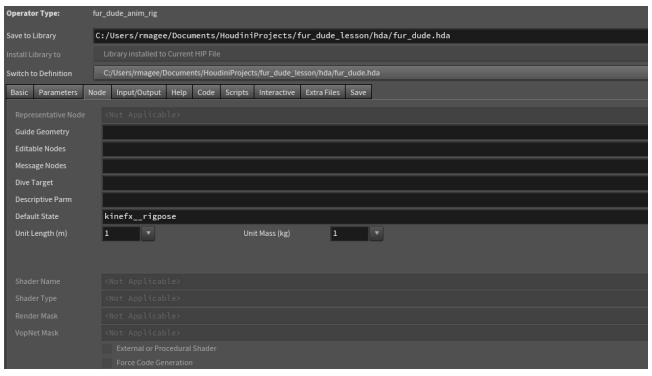
02

オブジェクトレベルに移動します。***furdude_rig*** ノードを **Alt ドラッグ**して新しいジオメトリノードを作成し、***test_rig*** と名前を付けます。T を押して、テストリグを左に動かします。

ダブルクリックして、このノードに入ります。***fur_dude_anim_rig*** ノードを選択し、**Assets** メニューから、**Lock Asset > Fur Dude Anim Rig** を選択します。

1 つ上のレベルに戻ります。これで 2 つのバージョンのリグができました。***test_rig*** はロックされ、完成したアセット内でどのように振舞うかを確認できます。現在は相互作用するものが何もありません。これは後ほど解決します。

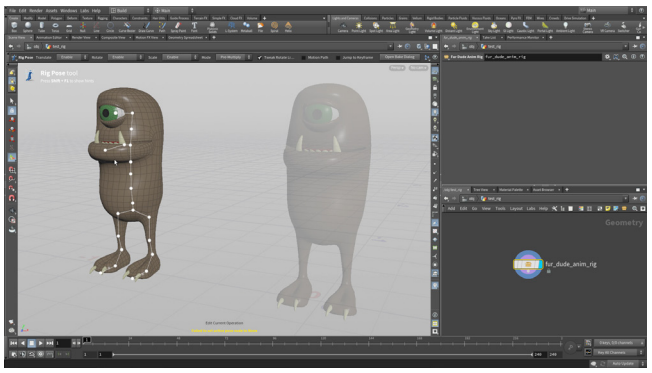
ネットワークエディタでクリックして、**Ctrl + 1** を押して Quick マークを設定します。



03

furdude_rig オブジェクトに移動したら、さらに ***fur_dude_anim_rig*** に戻ります。ネットワークエディタでクリックして、**Ctrl + 2** を押して Quick マークを設定します。これで、これらのネットワーク間を素早く行き来できるようになりました。

rigpose を選択し、**Transformations** の横にある **Clear** ボタンをクリックします。**Asset** メニューに移動し、**Save Asset > Fur Dude Anim Rig** を選択します。Asset メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。**Node** タブをクリックして、**Default State** を `kinefx__rigpose` に設定します。**Accept** をクリックします。



04

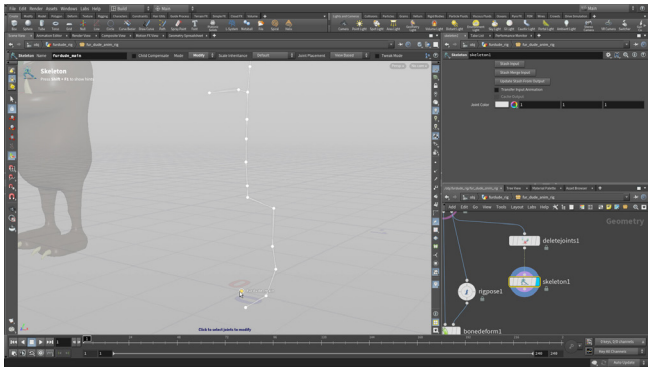
ネットワークビューで、**1 を押して test_rig** に戻ります。左側の Scene View で、ツールバーを展開して **Handle** ツールをクリックします。

これで、***test_rig*** にジョイントが表示されます。ジョイントをクリックしても変更は不可能です。これは、パラメータがアセットにプロモートされていないためです。ここから、このキャラクタにアニメーション可能なインターフェースを構築していきます。

パート 10

コントロールジョイントの追加

コントロールリグの柔軟性を高めるため、スケルトン全体のルートジョイント、リバースフットのセットアップ用にかかとジョイント、眼球のターゲットとなる Look At ポイントなど、複数のジョイントを追加します。これらのジョイントには、元の静止スケルトン内のジョイントと同じ名前を付けます。こうしておくと、キャラクタのモーションを駆動するために使用できます。

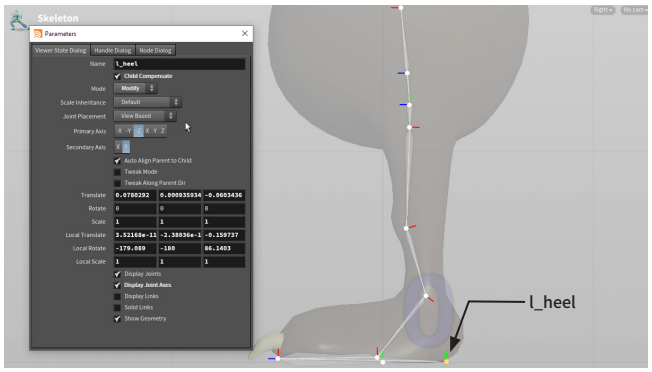


01

ネットワークビューで、**2 を押して fur_dude_anim_rig** に戻り、**Display フラグ**を設定します。***fur_dude_capture_rig*** ノードに **Template フラグ**を設定します。

Delete Joints ノードを ***fur_dude_capture_rig*** ノードの 3 つ目の出力から分岐させます。Group の横にある矢印をクリックして、ファー・デュードの右脚のジョイントを選択します。**Enter** を押します。

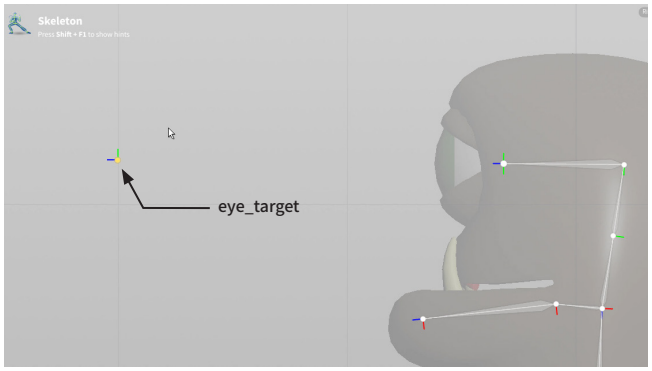
Skeleton ノードを追加します。**Mode** を **Create** に変更します。**グリッドスナッピング**をオンにして、原点にポイントを追加します。**Modify** モードに変更して、新しいジョイントの名前を ***furdude_main*** に変更します。



02

Right ビューに移動します。**Create** モードに戻り、**MMB クリック**してメインジョイントの選択を解除します。***l_toe*** ジョイントをクリックして、かかとがあるべき場所に新しいジョイントを追加します。

Modify モードに変更して、新しいジョイントの名前を ***l_heel*** に変更します。**P** を押して、**Child Compensate** を**オン**にし、**Rotate** を **0, 0, 0** に設定します。



03

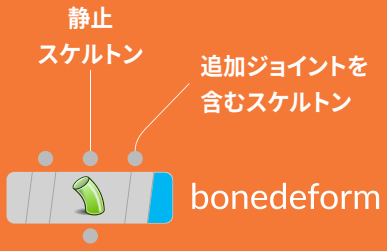
目の領域に移動します。**Create** モードに戻し、**MMB クリック**して現在の選択を解除します。目の前に新しいジョイントが追加されました。

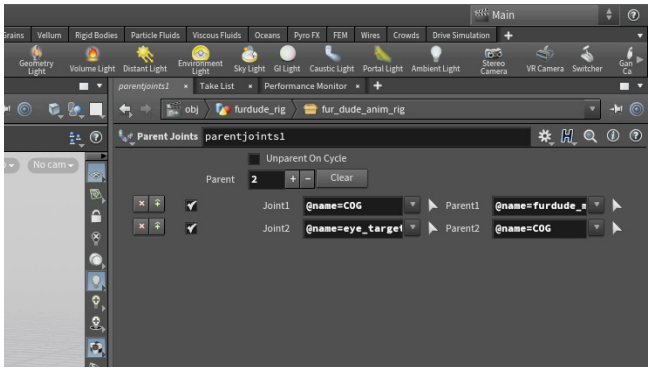
Modify モードに変更して、新しいジョイントの名前を ***eye_target*** に変更します。**P** を押して、**Rotate** を **0, 0, 0** に設定します。

リグにジョイントを追加する

これらの追加ジョイントは右端のノードストリームに追加されました。一方、中央の静止スケルトンは元のジョイントのままです。追加のジョイントを bonedeform ノードに接続しても、「ファントム」ジョイントは無視されます。元のジョイントだけが、リグの最終出力を決定します。

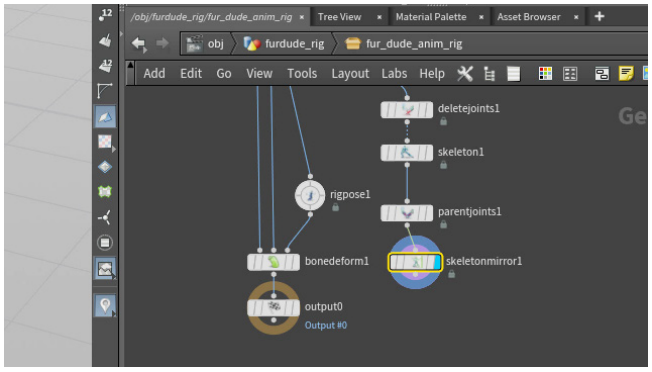
これらの追加ジョイントを bonedeform ノードの中央の静止スケルトン入力に接続すると、エラーになります。これらのジョイントは、入力ジオメトリに、対応するキャプチャウェイトを持っていないからです。





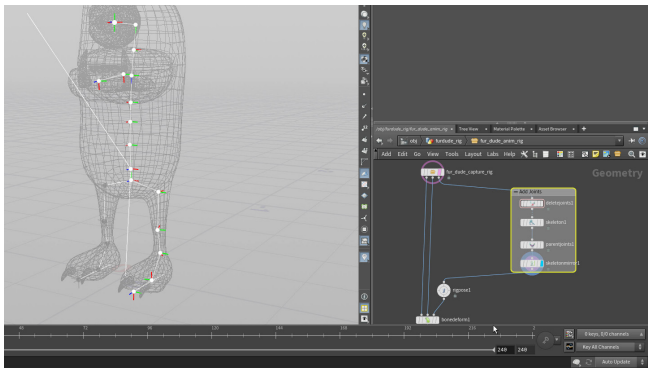
04 **Parent Joints** ノードを追加します。**+(プラス)記号**を 2 回クリックします。**Joint1** の横にある矢印をクリックして、Scene View で ***COG*** ジョイントをクリックします。(カーソルを Scene View に置いたまま) **Enter** を押して確定します。次に、**Parent1** の横にある矢印をクリックして、***furdude_main*** ジョイントを選択します。

2 つ目のエントリでは、**Joint2** の横にある矢印をクリックして、Scene View で ***eye_target*** ジョイントをクリックします。(カーソルを Scene View に置いたまま) **Enter** を押して確定します。次に、**Parent2** の横にある矢印をクリックして、***COG*** ジョイントを選択します。



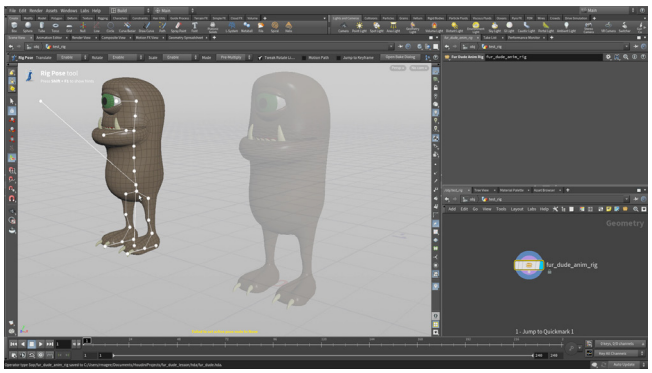
05 **Skeleton Mirror** ノードをチェーンに追加します。これにより、すべてのジョイントのミラーコピーが作成されます。パラメータエディタに移動して、**Group** の横の矢印をクリックします。左脚のジョイントのみを選択して (新しい ***l_heel*** ジョイントを含む)、**Enter** を押します。脚のみがミラー化されました。

Naming で、**Find Tokens** を ***l_***、**Replace Tokens** を ***r_*** に設定します。これで、右脚のジョイントに適切な名前が付けられます。



06 skeletonmirror ノードを ***rigpose*** ノードに接続します。***bonedeform*** ノードに **Display フラグ**を設定します。足のつめが反転しているように見えます。そのスケルトンジョイントに戻り、***l_toe*** を選択し、**P を押して**パラメータを表示します。**Rotate** を **0, 0, -90** に設定します。

ジョイントを追加するのに使用した 4 つのノードを選択し、**Add Network Box** ボタンをクリックします。ノードを囲むようにボックスを配置します。ボックスのタイトルをクリックして、***Add Joints*** と入力します。



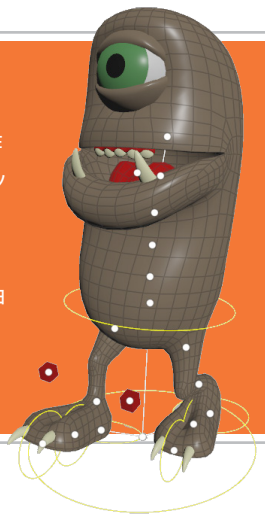
07 **Assets** メニューから、**Save Asset > Fur Dude Anim Rig** を選択します。変更がアセット定義に保存され、***test_rig*** が更新されます。まだ ***test_rig*** を編集することはできません。いずれのパラメータもトップレベルにプロモートされていないからです。

次は、メインコントロールをセットアップし、パラメータをプロモートして、***test_rig*** に生命を吹き込んでいきます。

テストリグの役割

現在作業しているリグは、ロックされていないアセットです。アセット内のすべてのジョイントを操作できます。ただし、アセットがアニメータに渡されたときにアニメータが扱えるのは、キャラクタのトップレベルにプロモートされたパラメータのみです。

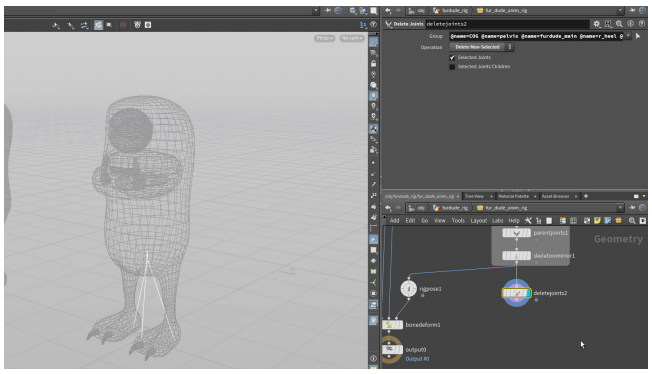
test_rig はアセットの 2 つ目のロックバージョンであり、パラメータをプロモートしたりコントロールを構築するまで、アセットを操作することはできません。テストリグが素晴らしいのは、アニメーションを付けられる状態になっているかどうかを確認できるツールだからです。テストリグを操作できなければ、アニメータはキャラクタにポーズを付けることができません。



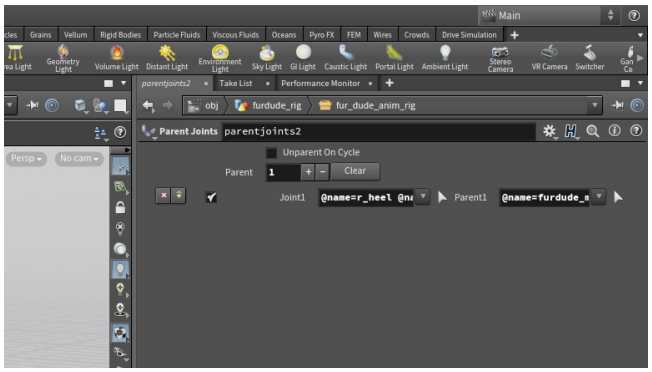
パート 11

メインコントロール

キネマティクスを追加するためには、COG の下に足がある、現在の階層を切り離す必要があります。いくつかのジョイントを切り離し、再度親子化して、希望通りの階層を構築します。Bone Deform を適切に動作させるには、この再親子化を別に行ってから、結果を元のスケルトン階層にブレンドすることが重要です。



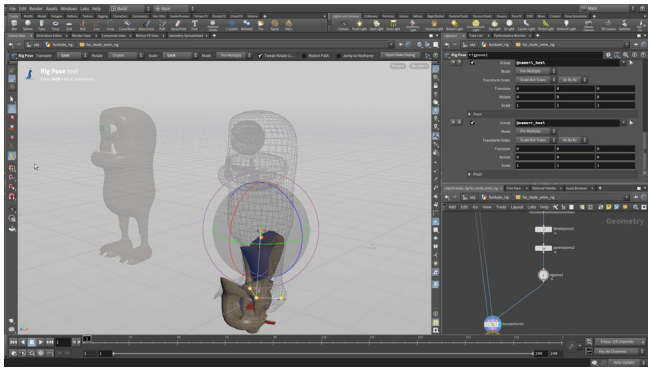
01 **Delete joints** ノードを ***skeletonmirror*** から分岐させます。Group の横にある矢印をクリックして、Scene View で ***furdude_main***、***COG***、***pelvis***、***l_heel***、***r_heel*** ジョイントを選択します。**Enter** を押し、**Operation** を **Delete Non-Selected** に設定します。



02 deletejoints ノードの後に **Parent Joints** ノードを追加します。**+(プラス)記号**をクリックして、ジョイントリストを追加します。**Joint1** の横にある矢印をクリックして、Scene View で 2 つの ***heel*** ジョイントを選択します。**Enter** を押します。

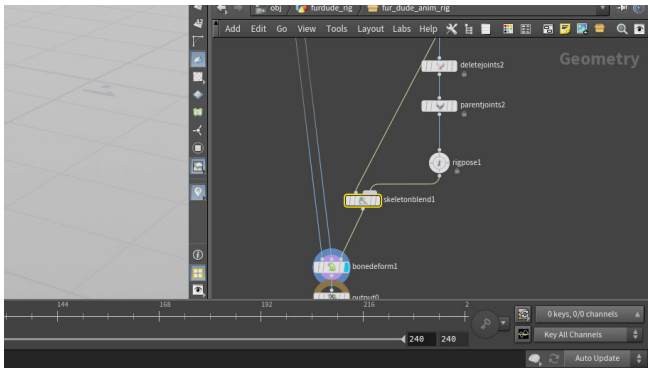
Parent1 の横にある矢印をクリックして、Scene View で ***furdude_main*** ジョイントを選択します。**Enter** を押します。

heel ジョイントが ***COG*** から切り離され、脚の下部で IK を駆動するのに使用できるようになりました。***COG*** ジョイントに親子化されている ***pelvis*** ジョイント以外、すべてのジョイントは ***furdude_main*** ジョイントに親子化されています。



03 ***parentjoints*** ノードを ***rigpose*** ノードに接続します。このノードからの既存のジョイントをすべてクリアして、**Display フラグ**を設定します。***furdude_main*** ジョイント、***COG*** ジョイント、***pelvis*** ジョイント、2 つの ***heel*** ジョイントをクリックします。

ここで ***bonedeform*** ノードに **Display フラグ**を設定すると、ジョイントの多くがなくなっているため、想像とは違う表示になります。これは、Skeleton Blend を使用して修正できます。



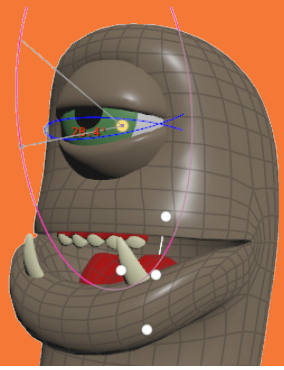
04 ネットワークエディタに **Skeleton Blend** ノードを追加します。***skeletonmirror*** ノードを左側の入力に、***rigpose*** ノードを右側の入力に接続します。その後、***skeletonblend*** ノードを ***bonedeform*** ノードの 3 つ目の入力に接続します。

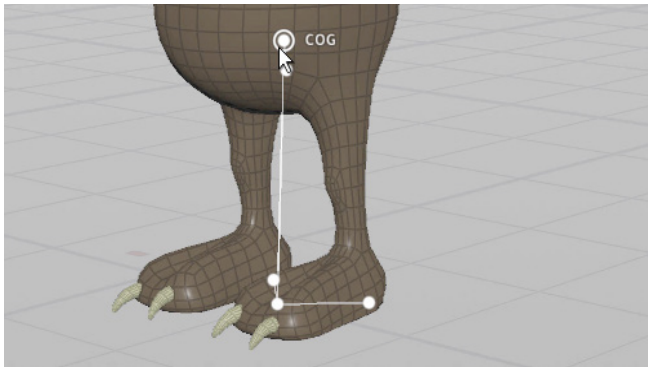
skeletonblend のパラメータエディタで、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** にします。

コントロールジオメトリ

ジョイントを rigpose に追加するとき、アセットのトップレベルにプロモートすると、ジョイントを選択してアニメートできるようになります。**Attach Control Geometry** ノードを使って、ジオメトリをリグのさまざまなパーツに割り当てれば、ジョイントの選択や操作が簡単になります。好きな形状を作成して、コントロールに使用できます。

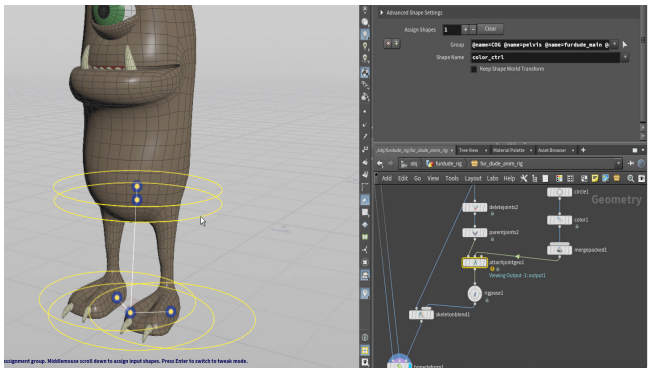
コントロールジオメトリの用途としては、眼球や、重なり合った 2 つの lid（まぶた）ジョイントなどが挙げられます。コントロールジオメトリを使用すれば、それらのジョイントを簡単に選択して、その部分のリグをセットアップできます。ここでは、メインコントロールに使用します。





05 **Assets** メニューから **Save Asset > Fur Dude Anim Rig** を選択します。これにより現在の設定が保存されます。ネットワークビューで、**1 を押して test_rig** に戻ると、rigpose にリストされている 5 つのジョイントのみが表示されます。

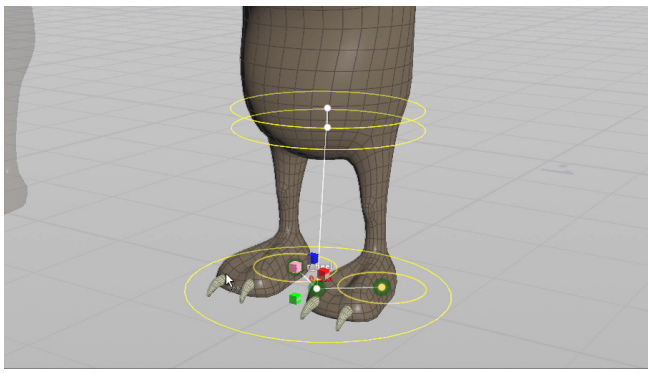
パラメータはまだプロモートされていないため、ジョイントの選択や移動はできません。



06 ネットワークビューで、**2 を押して fur_dude_anim_rig** に戻ります。**Circle** ノードをネットワークに追加します。**Orientation** を **ZX** に設定します。**Uniform Scale** を **0.2** に設定します。**Divisions** を **36**、**Arc Type** を **Open Arc** に設定します。**Color** ノードを追加して、**Color** を**黄色**に設定します。

さらに **Merge Packed** ノードを追加して、**Name 1** を ***circle_ctrl*** に設定します。

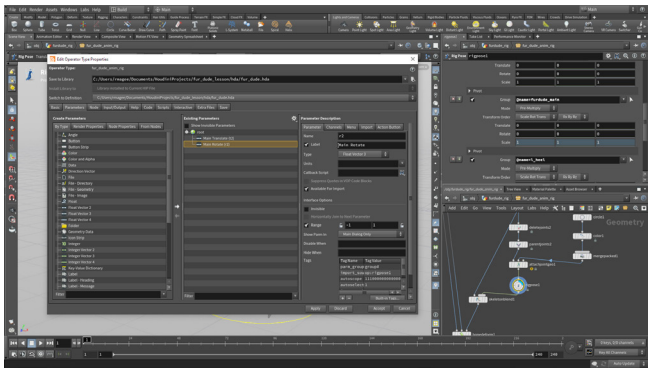
parent ノードと ***rigpose*** ノードの間に **Attach Joint Geometry** ジオメトリを配置します。その 2 つ目の入力に ***mergepacked*** ノードを接続します。このノードに **Display フラグ**を設定し、Scene View で **Enter** を押します。表示されているジョイントをすべて選択したら、**G を押し**、スクロールホイールを使用して ***circle_ctrl*** ジオメトリを見つけます。ジオメトリがすべてのジョイントに割り当てられます。



07 上部のオペレーションコントロールツールバーで、**Mode** を **Tweak Shapes** に変更します。

COG と ***pelvis*** ジョイントを選択し、**G を押して**トランスフォームハンドルを表示します。**E を押して**スケールハンドルにしたら、中央のハンドルをクリック & ドラッグして、これらのコントロールが少し小さくなるまで 3 方向すべてでスケールします（パラメータエディタで約 0.67）。

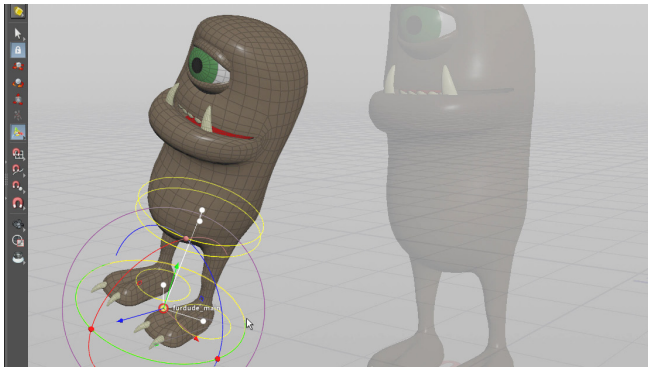
2 つの ***heel*** ジョイントを選択し、**G を押して**トランスフォームハンドルを表示します。**E を押して**スケールハンドルにしてから、中央のハンドルをクリック & ドラッグして、かかとのコントロールがかなり小さくなるまで 3 方向すべてでスケールします（パラメータエディタで約 0.3）。



08 ***rigpose*** ノードを選択し、コントロールジオメトリを使用して 4 つのジョイントを選択します。これをテストリグで動作させるには、パラメータをプロモートする必要があります。

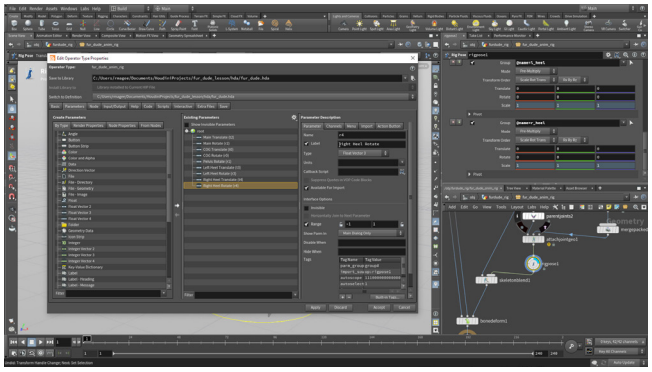
Assets メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。**Parameters** タブをクリックします。

rigpose ノードで、***furdude_main*** に移動し、**Scale** で **RMB クリック**して **Lock Parameter** を選択します。**Translate** をドラッグして、***root*** の下のパラメータリストに移動します。**Label** を ***Main Translate*** に設定します。**Rotate** パラメータも同じようにして、***Main Rotate*** と名前を付けます。**Accept** をクリックして終了し、結果をアセットに保存します。



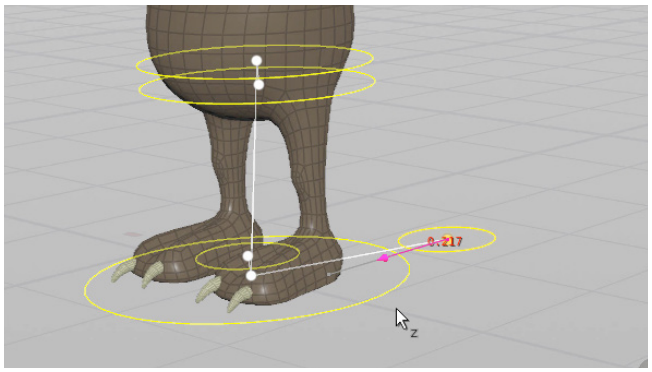
09 ネットワークビューで、**1 を押して test_rig** に戻ります。テストリグが更新され、新しいコントロールが表示されています（**Handle** ツールがアクティブになっている場合）。コントロールジオメトリを使用して ***furdude_main*** ジョイントを選択すると、トランスフォームハンドルが表示され、それを使ってリグを動かすことができます。

動かした後は**取り消し**て、元の位置に戻します。



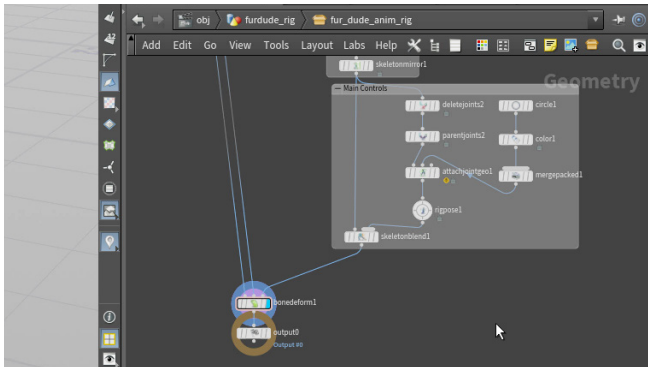
10 ネットワークビューで、**2 を押して fur_dude_anim_rig** に戻ります。**Assets** メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。***COG***、***l_heel***、***r_heel*** ジョイントの **Translate** と **Rotate**、pelvis ジョイントの Rotate をドラッグします。いずれの場合も、各ジョイントの使用しない **Scale** や **Translate** パラメータ（pelvis の場合）はロックします。

Accept をクリックして終了し、結果をアセットに**保存**します。



11 ネットワークビューで、**1 を押して test_rig** に戻ると、新しいコントロールが表示されるようになりました。コントロールジオメトリを使って ***COG*** ジョイントと ***heel*** ジョイントを選択し、パーツをトランスフォームさせてみましょう。

終了したら**取り消し**て、すべてのパーツを元の位置に戻します。



12 ネットワークビューで、**2 を押して fur_dude_anim_rig** に戻ります。メインコントロールのセットアップに使用したノードを選択し、**Add Network Box** ボタンをクリックします。ノードを囲むようにボックスを配置します。ボックスのタイトルをクリックして、***Main Controls*** と入力します。

Assets メニューから、**Save Asset > Fur Dude Anim Rig** を選択します。この操作はリグに影響しませんが、アセットを最新の状態に保つことができます。また、シーンファイルも**保存**しましょう。

ネットワークの整理

ひと手間かけて、ノードを整列させたり、**ネットワークボックス**を追加しておきましょう。よく整理されたネットワークは、後で作業しやすく、他のスタッフにも意図が伝わりやすいものです。

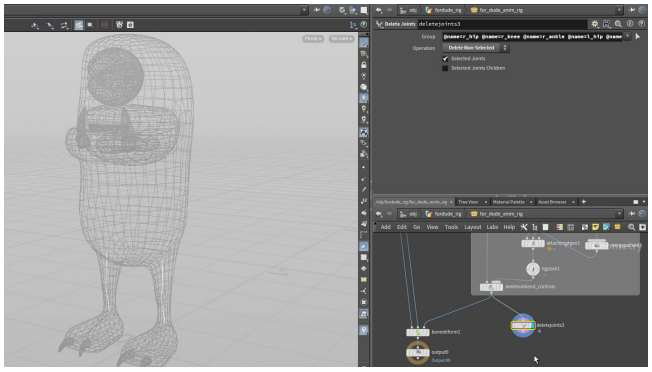
また、各ノードにはコメントを追加して、ネットワークに表示させることができます。**ステッキーノート**を使用すれば、ノードの大きいブロックについての説明も付加できます。チームでネットワークを作成する場合には、コミュニケーションが常に重要です。

This part of the network organizes the main controls such as the root, the COG and the heel joints.

パート 12

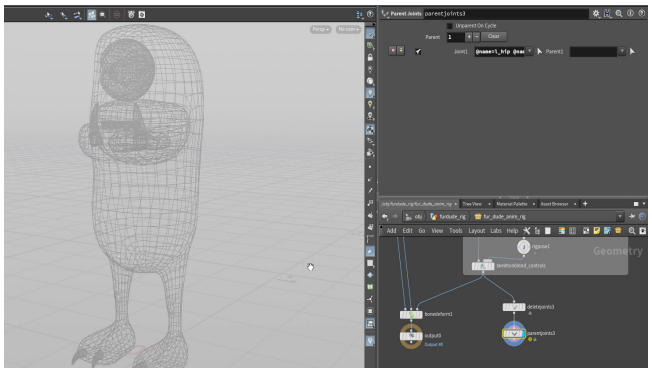
脚のインバースキネマティクス

キャラクタのアニメーションでは、脚にインバースキネマティクスを設定して、足あるいは腰の動きに合わせて膝が適切に曲がるようにできます。このパートでも、メインのスケルトンのジョイントを利用し、KineFX を使ってセットアップしていきます。結果も、前のパートと同じようにオリジナルの階層にブレンドして戻します。

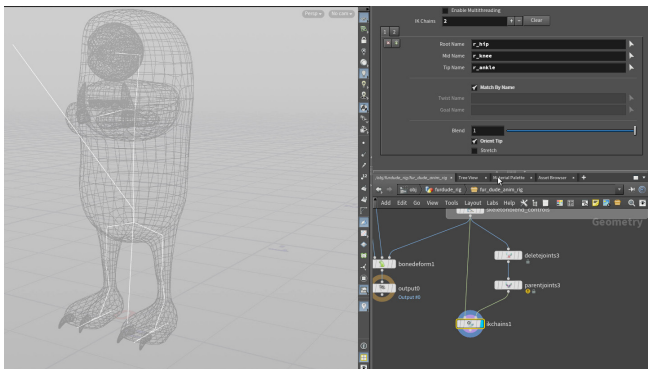


01 ***skeletonblend*** ノードの名前を ***skeletonblend_controls*** に変更します。これらのノードはよく使用するので、識別しやすくしましょう。

skeletonblend_controls ノードから **Delete Joints** ノードを分岐させ、**Display フラグ**を設定します。Group の横の矢印をクリックし、Scene View で左右の ***hip***、***knee***、***ankle*** の各ジョイントを選択します。**Enter** を押し、**Operation** を **Delete Non-Selected** に設定します。

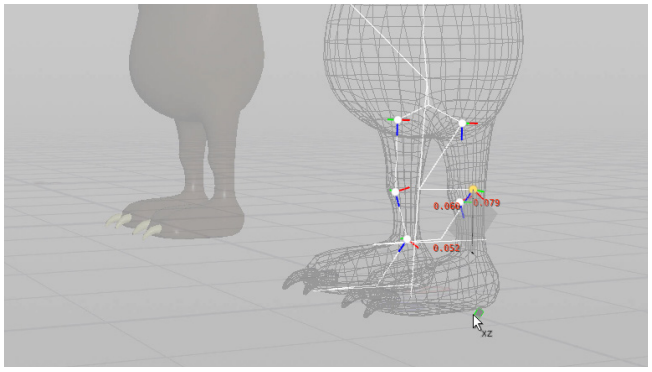


02 **Parent Joints** ノードを分岐させ、**Display フラグ**を設定します。**+（プラス）記号**をクリックしてジョイントを追加し、**Joint1** を * に設定します。**Parent1** は空のままにします。これで、すべてのジョイントの接続が解除されました。各ジョイントを個別に使用できます。



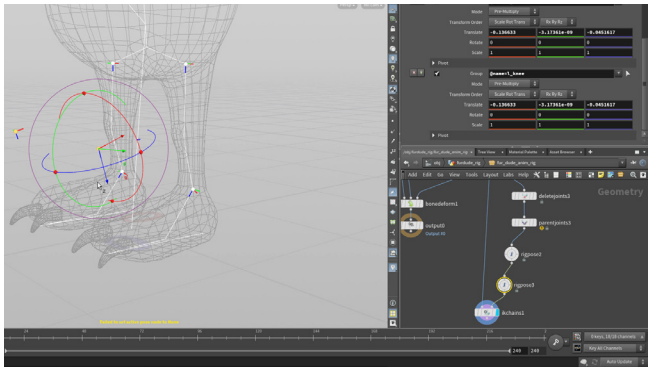
03 **IK Chains** ノードをネットワークエディタに配置します。その 1 つ目の入力に ***skeletonblend_controls*** を接続し、2 つ目の入力に ***parentjoints*** ノードを接続します。**Display フラグ**を設定します。

パラメータエディタで **+（プラス）記号**をクリックし、**Root Name** の横の矢印をクリックします。***l_hip*** ジョイントを選択して、（Scene View にカーソルを置いたまま）Enter を押します。**Mid Name** を ***l_knee***、**Tip Name** を ***l_ankle*** に設定します。矢印を使用してジョイントを選択してもよいですし、名前を入力してもかまいません。**Match by Name** を**オン**にし、**Blend** を **1** に設定したら、**Orient Tip** を**オン**にします。**+（プラス）記号**をもう一度クリックして、右脚についても同じ手順を繰り返します。



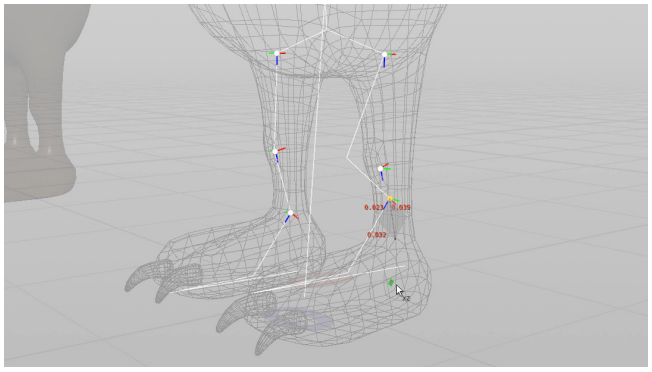
04 **Rig Pose** ノードを ***parent*** ノードと ***ikchains*** ノードの間に追加します。左右の ***ankle*** ジョイントをクリックして動かし、インバースキネマティクスが機能していることを確認します。**Match by Name** をオンにしたので、ankle ジョイントは、IK チェーンのエンドエフェクタとして動作しています。

足首を動かすと、時々、膝が逆方向に曲がります。これは、knee ジョイントがツイストエフェクタとして使用され、適切に配置されていないことが原因です。脚の前方に移動するようにしなければなりません。

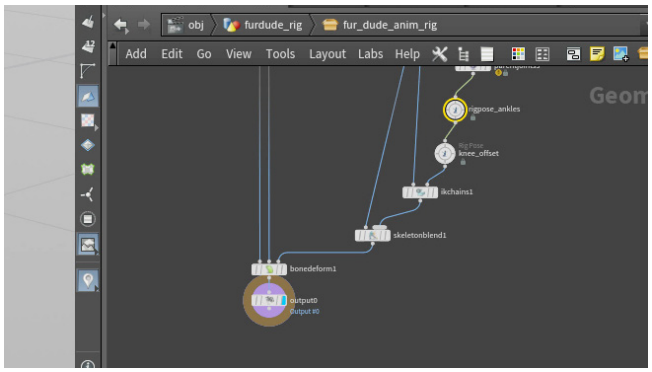


05 **Rig Pose** ノードを最初の ***rigpose*** ノードと ***ikchains*** ノードの間に追加します。**Shift** キーを押しながら両方の ***knee*** ジョイントを選択し、それらをキャラクタの前方に動かします。これで、IK での膝の向きは正常になるはずです。

1 つ目の rigpose ノードの名前を ***rigpose_ankles***、2 つ目の rigpose ノードの名前を ***knee_offset*** にします。

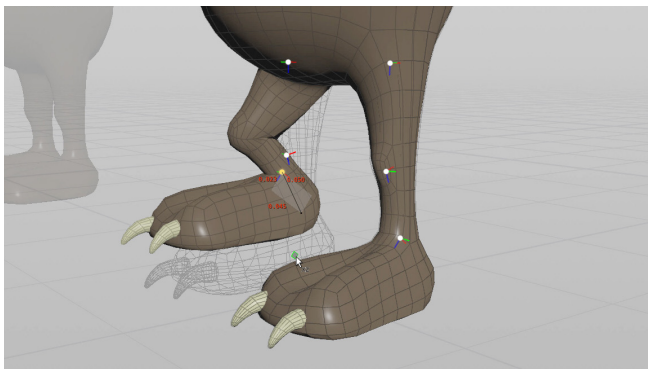


06 ***rig_ankles*** ノードに戻り、左右の ankle ジョイントをテストします。膝をオフセットしたので、逆方向に曲がることはありません。また、***knee_offset*** ノードを選んで膝を動かし、IK チェーンのツイストエフェクタとして動作していることも確認します。



07 **Skeleton Blend** ノードを追加して、1 つ目の入力に ***skeletonblend_controls*** ノード、2 つ目の入力に ikchains ノードを接続します。この新しい **Skeleton Blend** ノードの出力を Bone Deform ノードの 3 つ目の入力に接続します。

このノードの名前を ***skeletonblend_ik*** に変更し、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** に設定します。

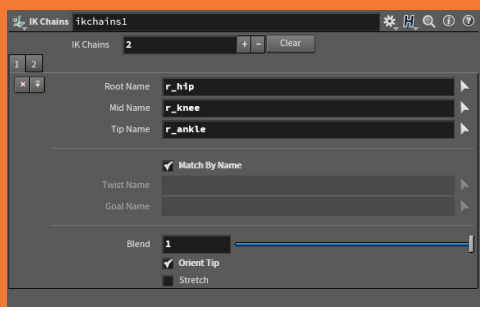


08 ***rigpose_ankles*** ノードに戻り、ankle ジョイントがキャプチャしたサーフェスに作用するかどうかをテストします。

この時点では、デジタルアセットを保存して変更をテストリグにプッシュする操作は行いません。最終的なリグでは、足首のコントールに ***rigpose_ankles*** ノードは使用しないからです。以降の手順でリバースフットのセットアップを構築し、足全体のセットアップを機能させてから、脚全体のセットアップをアセットに保存します。

FK/IK のブレンド

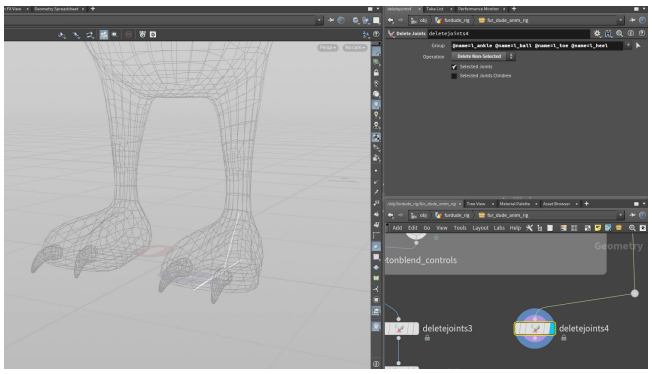
IK と FK をブレンドする方法はいくつかありますが、このレッスンでは使用しません。このキャラクタの脚には IK しか使わないからです。ブレンドするには、rigpose を使用しているアセットに脚のジョイントのパラメータをプロモートしてから、IK Chains ノードの **Blend** アトリビュートを使用するか、IK ソリューションと rigpose の回転ジョイントの間で **Skeleton Blend** を使用します。ファー・デュードに腕があれば、このセットアップをすることになるでしょう。



パート 13

リバースフットのセットアップ

足をコントロールするために、伝統的なリバースフットのセットアップを作成します。かかと（heel）がルートで、つま先（toe）、ボール（ball: 土踏まずから前）、足首（ankle）の親になります。KineFX なら簡単に作成でき、元のスケルトンにブレンドできます。ここでは右足を完全に再構築しますが、ジョイント名が揃っているので、すべて正常に機能するはずです。

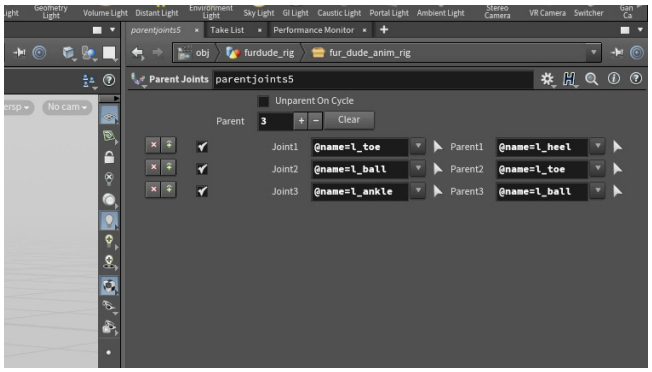


01

rigpose_ankles ノードを削除します。足首は、リバースフットのセットアップを使用してコントロールします。

skeletonmirror ノードから **Delete Joints** ノードを分岐させ、**Display フラグ**を設定します。Group の横の矢印をクリックし、Scene View で ***l_ankle***、***l_ball***、***l_toe***、***l_heel*** ジョイントを選択します。**Enter** を押し、**Operation** を **Delete Non-Selected** に設定します。

注：ワイヤーを迂回させて、ネットワークボックスの周りを通るようにしたいときは、ワイヤーを **Alt クリック**してドットを追加します。

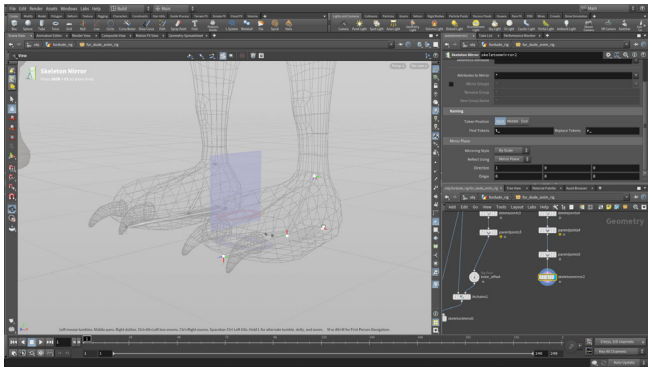


02

Parent Joints ノードを追加し、**Display フラグ**を設定します。**+（プラス）記号**をクリックしてジョイントを追加し、**Joint1** を * に設定します。**Parent1** は空のままにします。

2 つ目の **Parent Joints** ノードを分岐させ、**Display フラグ**を設定します。Scene View で ***l_heel*** ジョイント、***l_toe*** ジョイント、***l_ball*** ジョイント、***l_ankle*** ジョイントとクリックします。MMB クリックして終了します。パラメータエディタでは、これらのジョイントが次の順番で表示されます。

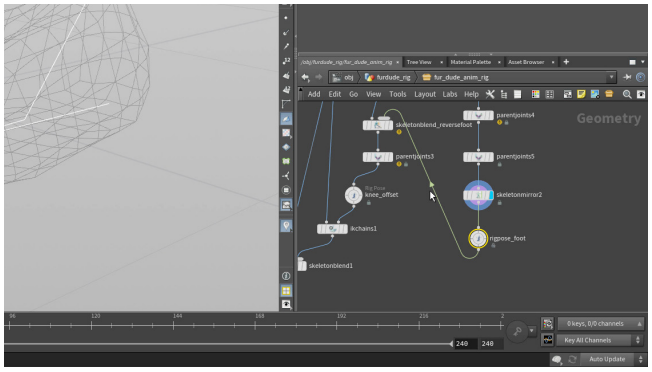
- **Joint1** : ***@name=l_toe*** | **Parent1** : ***@name=l_heel***
- **Joint2** : ***@name=l_ball*** | **Parent2** : ***@name=l_toe***
- **Joint3** : ***@name=l_ankle*** | **Parent3** : ***@name=l_ball***



03

Skeleton Mirror ノードをチェーンに追加します。パラメータエディタの Naming で、**Find Tokens** を ***l_***、**Replace Tokens** を ***r_*** に設定します。これで、右脚のリバースフットを作成できます。

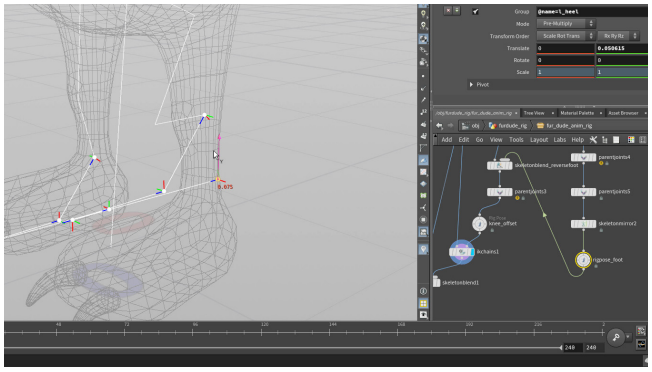
この階層内のすべてのジョイントは、オリジナルのスケルトンと同じ名前です。このためリグのポーズを設定する際にも、情報が正しく転送されます。



04

脚にセットアップした ***deletejoints*** と ***parentjoints*** の間に **Skeleton Blend** ノードを追加します。このノードの名前を ***skeletonblend_reversefoot*** に変更し、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** に設定します。**Group** の横の矢印をクリックし、左右の ***ankle*** ジョイントを選択します。**Enter** を押します。

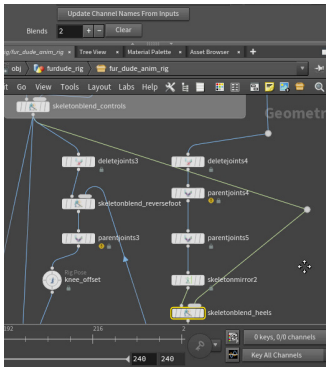
skeletonmirror の後に **Rig Pose** ノードを追加します。***rigpose_foot*** に名前を変更します。次に、***skeletonblend_reversefoot*** ノードの 2 つ目の入力に接続します。

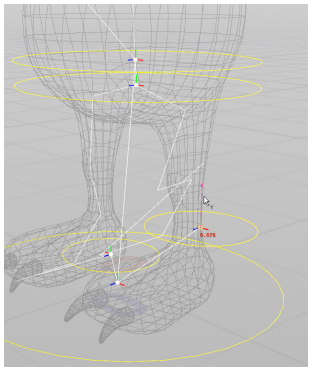


05

ikchains ノードに **Display フラグ**を設定します。***rigpose_foot*** ノードに移動し、***l_heel*** ジョイントを選択します。そのジョイントを動かすと、足と脚のチェーン全体が動きます。***l_toe*** ジョイントを選択して、回転します。同じようにリバースフットが機能し、IK チェーンがアクティブになります。

最後は **Clear** を押して、すべてのジョイントを削除します。いくつかは後でまた追加します。かかとのジョイントについては、メインコントロールの一部として前にセットアップした、heel ジョイントを使用します。

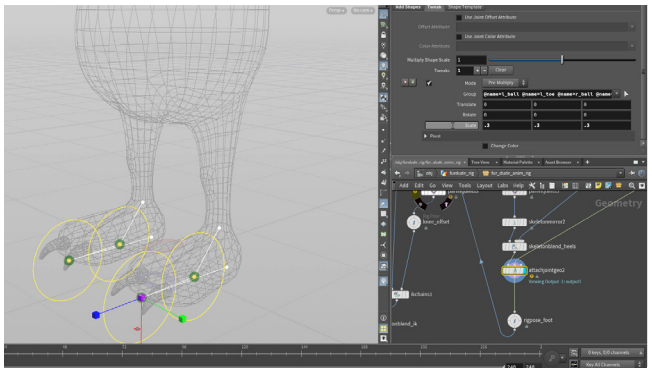




06

つい先ほど使用したリバースフットの ***skeletonmirror*** と ***rigpose*** ノードの間に **Skeleton Blend** ノードを追加します。このノードの名前を ***skeletonblend_heels*** に変更し、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** に設定します。その 2 つ目の入力に ***skeletonblend_controls*** ノードの出力を接続します。

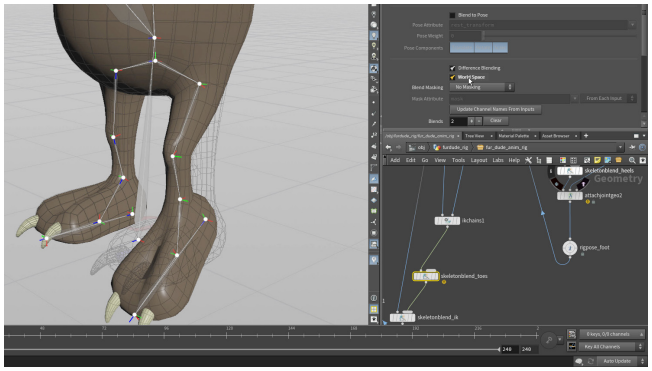
Group の横の矢印をクリックし、2 つの ***heel*** ジョイントを選択します。**Enter** を押します。メインコントロールの ***rigpose*** を選択したら、左または右のかかとを動かして、セットアップ全体をコントロールしてみましょう。COG ジョイントを選択して上下に動かしても、IK チェーンは適切に機能するはずです。



07

Attach Joint Geometry ノードを ***skeletonblend_heels*** ノードと ***rigpose_foot*** ノードの間に配置します。その 2 つ目の入力に、**Main Controls** ネットワークボックスの ***mergepacked*** ノードを接続します。このノードに **Display フラグ**を設定し、Scene View で **Enter** を押します。

toe と ***ball*** ジョイントを選択し、**G を押して**スクロールホイールを使用して ***circle_ctrl*** ジオメトリを見つけます。上部のオペレーションコントロールツールバーで、**Mode** を **Tweak Shapes** に変更します。***toe*** と ***ball*** ジョイントを選択し、**G を押します**。**E を押してスケール**ハンドルにしたら、中央のハンドルをクリック & ドラッグして、コントロールを約 **0.3** にスケールします。



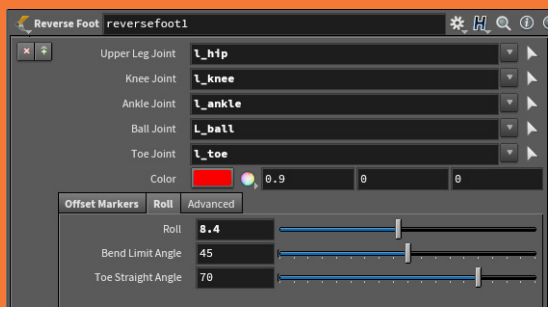
08

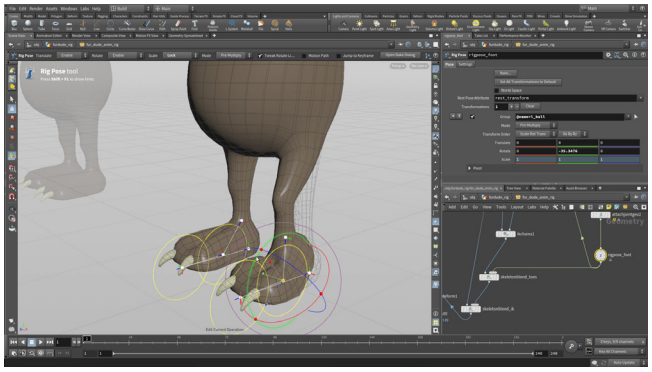
bonedeform ノードに **Display フラグ**を設定します。***rigpose_foot*** ノードを使用して ***l_ball*** のポーズを決めます。ボール（足の土踏まずから前）を回転させると、つま先は曲がるのではなく、下を向きます。

REVERSE FOOT SOP

Reverse Foot SOP は、足のコントロールや脚のキネマティクスの駆動に利用できます。**Reverse Foot SOP** には足の回転を制御できるスライダや、部位別のコントロールがあります。ただし、このレッスンでは使用しません。

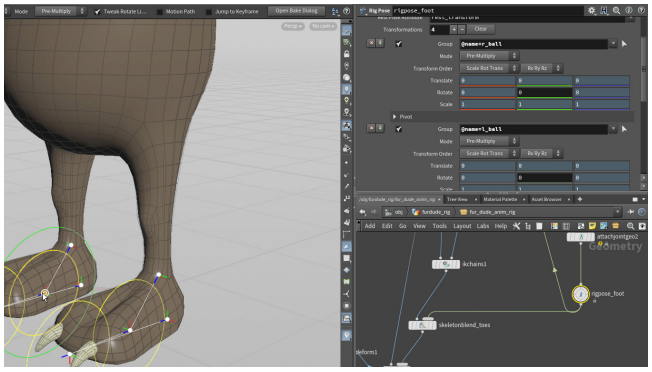
ここでは手作業でリバースフットソリューションを作成して、ジョイントをどのように操作すれば必要なコントロールが得られるかを学びます。





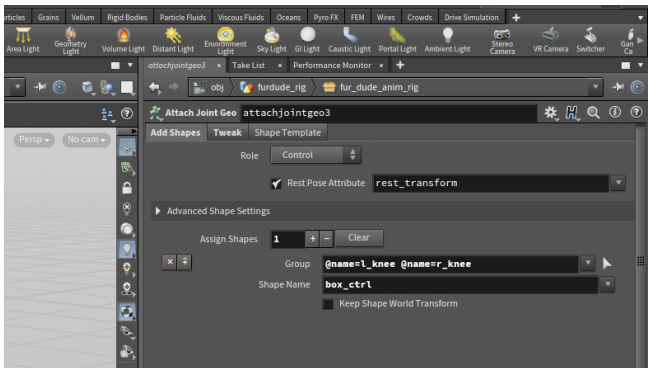
09 ***ikchains*** と ***skeletonblend_ik*** ノードの間に **Skeleton Blend** ノードを追加します。このノードの名前を ***skeletonblend_toes*** に変更し、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** に設定します。その 2 つ目の入力に ***rigpose_foot*** ノードの出力を接続します。**Group** の横の矢印をクリックし、***toe*** と ***ball*** ジョイントを選択します。**Enter** を押します。

これで ***ball*** ジョイントを回転すると、つま先が正しい方向を向くようになりました。



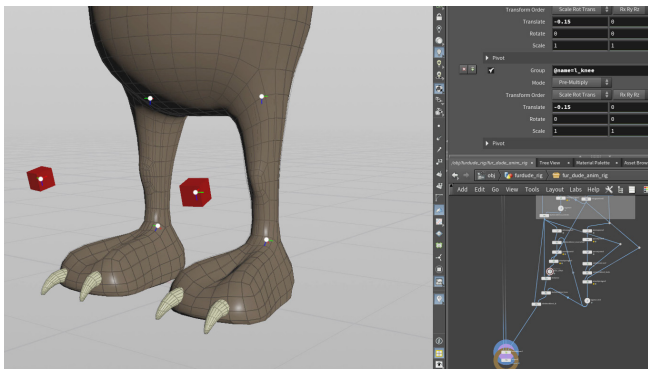
10 ***rigpose_feet*** ノードを選択します。このノードには***左右の ball*** と ***toe*** のジョイントのみがリストされていることを確認します。すべてのジョイントに対して、**Translate**、**Rotate**、**Scale** パラメータ上で **RMB クリック > Lock Parameter** を選択します。その後、それぞれのジョイントの **Rotate Y** で **RMB クリック > Unlock Parameter** を選択します。

これで、これら 4 つのジョイントのいずれかを選択したときに、Rotate Y のみが使用可能になりました。



11 Main Controls ネットワークボックスに戻ります。**Box** ノードをネットワークに追加します。**Uniform Scale** を **0.02** に設定します。次に **Color** ノードを追加して、**赤色**に設定します。このノードを ***mergepacked*** ノードに接続して、**Name 2** を ***box_ctrl*** に設定します。

parentjoints ノードと ***knee_offset*** ノードの間に **Attach Joint Geometry** ノードを配置します。その 2 つ目の入力に ***mergepacked*** ノードを接続します。**Mode** を **Tweak Shapes** に設定したら、***knee*** ジョイントを選択します。**G を押して**スクロールホイールを使用し、***box*** ジオメトリを見つけます。



12 ***knee_offset*** ノードに移動します。2 つの knee ジョイントのみがリストされていることを確認します。その両方のジョイントに対して、**Rotate** と **Scale** パラメータ上で **RMB クリック > Lock Parameter** を選択します。この 2 つのジョイントは、移動によってコントロールできるようになりました。

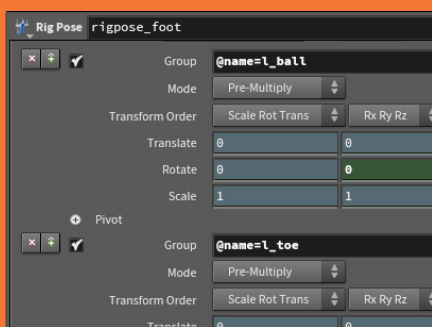
両方の膝の **Translate** を **-0.15, 0, -0.05** に設定します。

背骨と頭部のジョイントを定義するのに使用したこれらすべてのノードを囲む**ネットワークボックス**を追加し、***Leg Controls*** と名前を付けます。

RIG POSE の役割

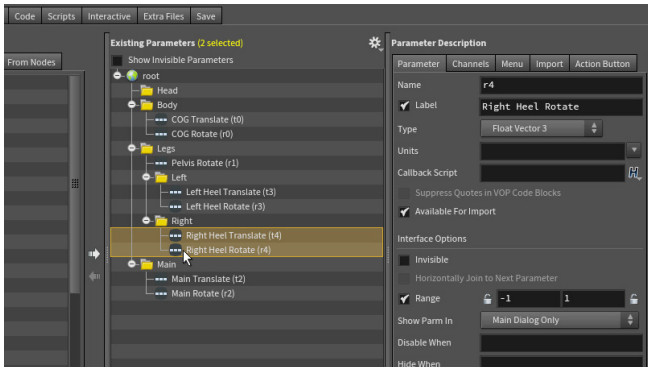
ここまで、Rig Pose ノードを使用してリグをテストしてきました。アニメーションコントロールリグを構築するときは、このノードを使って、トップレベルにプロモートされるパラメータをセットアップしたり、デジタルアセットの使用時に表示されるジョイントを定義することもできます。

こうしたノードのセットアップでは、不要なパラメータを誤って追加して、トップレベルに余計なジョイントが追加されてしまうことがよくあります。また、X ボタンを間違えてクリックして、必要なパラメータを削除してしまう可能性もあります。



パート 14
脚と背骨のコントロールのプロモート

脚と背骨のすべてのコントロールをアニメータが利用できるようにするには、パラメータをアセットのトップレベルにプロモートする必要があります。この手順を行わないと、アニメータに必要なコントロールを渡すことができません。また、アニメータに使用してほしくないパラメータを非表示にしておくことも可能です。

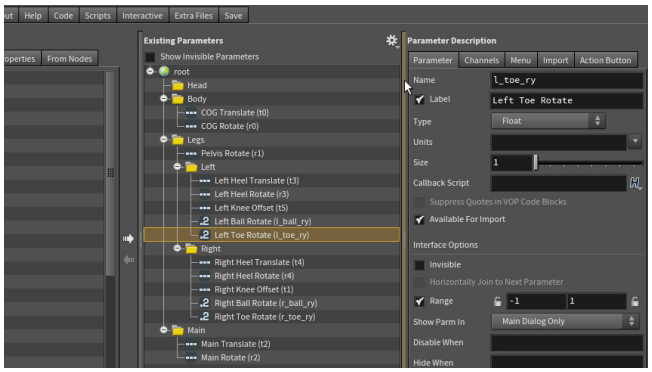


01

Assets メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。**Parameters** タブをクリックします。

フォルダを 4 つ作成します。それぞれ ***Head***、***Body***、***Legs***、***Main*** と名前を付けます。***COG パラメータ***を **Body** フォルダ、***Main パラメータ***を **Main** フォルダにドラッグします。

Legs フォルダの中に 2 つのフォルダを作成し、それぞれ **Left** および **Right** と名前を付けます。***Left Heel*** パラメータを **Left** フォルダに、***Right Heel*** パラメータを **Right** フォルダに追加します。

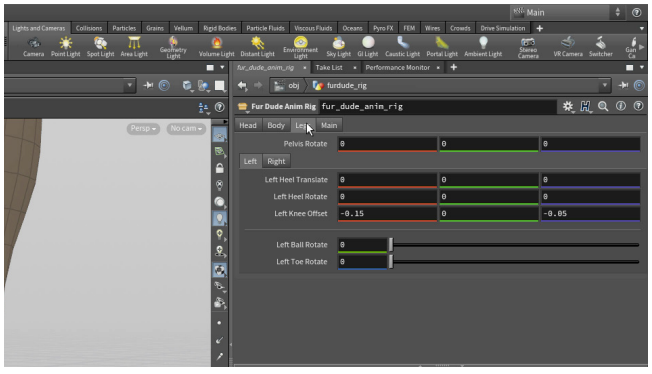


02

knee_offset ノードから、***l_knee*** の Translate 値を **Leg** フォルダの **Left** フォルダにドラッグします。名前を **Left Knee Offset** に変更します。**Channel** タブをクリックして、デフォルトを **-0.15, 0, -0.1** に設定します。

rigpose_foot ノードで、***l_ball*** ジョイントから **Rotate Y** をドラッグし、名前を **Left Ball Rotate** に変更して、**Range** を **0, 30** に設定します。同様に ***l_toe*** ジョイントから **Rotate Y** をドラッグし、名前を **Left Toe Rotate** に変更して、**Range** を **0, 20** に設定します。***Knee Offset*** と ***Ball Rotate*** パラメータの間に **Separator** をドラッグします。

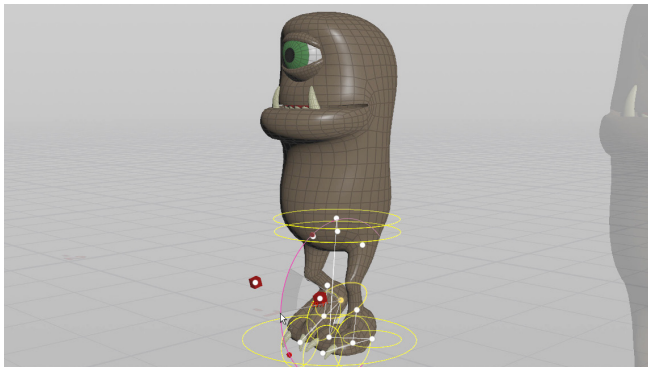
右膝および右足についても同じ手順を繰り返します。



03

Accept をクリックし、新しいパラメータとコントロールをアセットに保存します。1 つ上のレベルに移動して、***furdude_rig*** にレイアウトされたアセットパラメータを確認します。

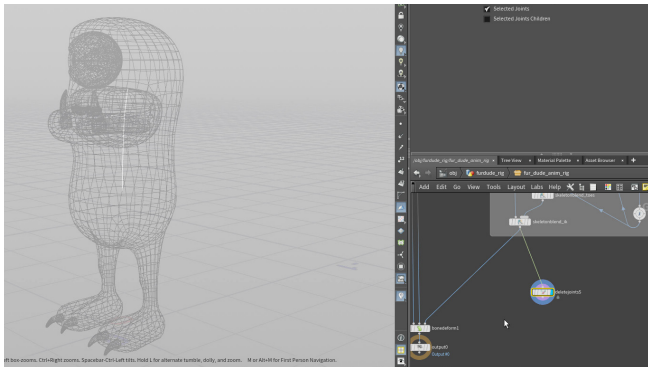
これらのパラメータは、作業対象のロックしていないリグに設定されています。これらのパラメータをそのままテストするのではなく、***test_rig*** を使ってテストしましょう。



04

ネットワークビューで、**1 を押して *test_rig*** に戻ると、新しいコントロールが表示されるようになりました。

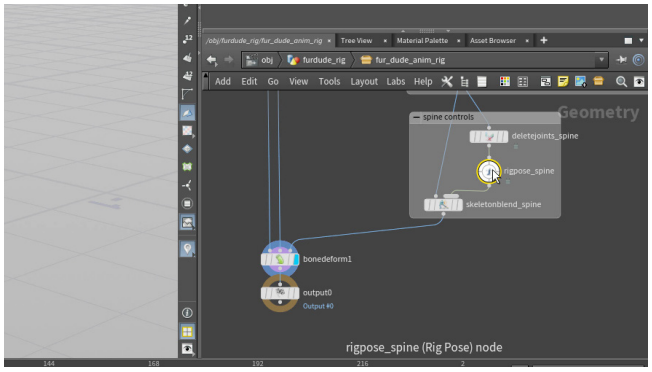
Scene View のテストリグのさまざまな動作を、ハンドルやフローティングパネルのパラメータを使って試します。



05

skeletonblend_ik ノードから **Delete Joints** ノードを分岐させて、その **Display フラグ**を設定します。名前を ***deletejoints_spine*** に変更します。

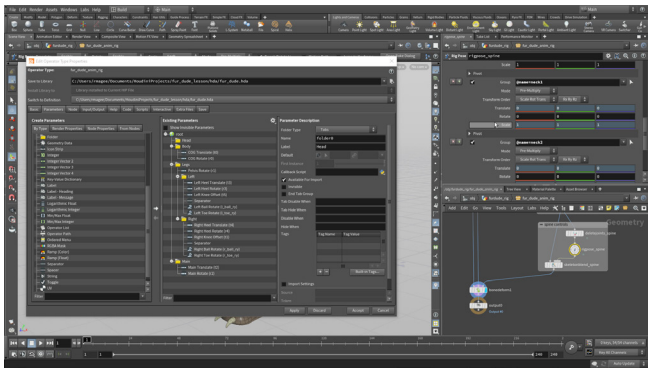
Group の横の矢印をクリックし、Scene View で ***spine1***、***spine2***、***spine3***、***neck1***、***neck2***、***jaw*** ジョイントを選択します。**Enter** を押し、**Operation** を **Delete Non-Selected** に設定します。



06

Rig Pose ノードを追加して、名前を ***rigpose_spine*** に変更します。***skeletonblend_ik*** と ***bonedeform*** ノードの間に **Skeleton Blend** ノードを追加します。ノードの名前を ***skeletonblend_spine*** に変更し、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** に設定します。その 2 つ目の入力に ***rigpose_spine*** ノードの出力を接続します。

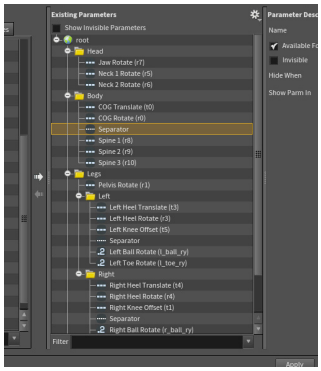
背骨と頭部のジョイントを定義するのに使用したこれらすべてのノードを囲む**ネットワークボックス**を追加し、***Spine Controls*** と名前を付けます。

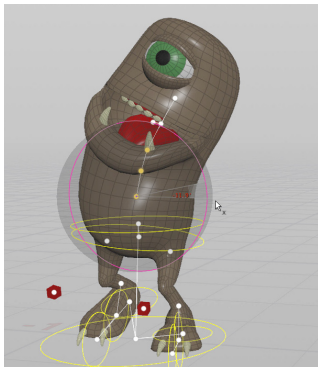


07

Display フラグを設定したら、Scene View で **S キー**を押しながらすべてのジョイントを選択します。これらのジョイントがリグポーズのリストに追加されます。

これらすべてのジョイントについて、**Translate** および **Scale** パラメータをロックします。ここでは回転のみを使用します。





08

Assets メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。**Parameters** タブをクリックします。

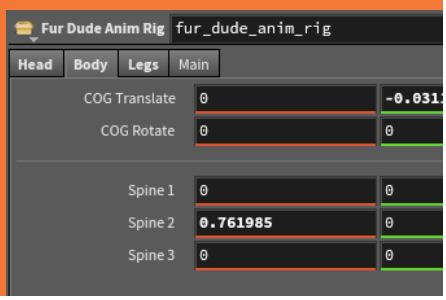
neck1、***neck2***、***jaw*** を **Head** フォルダにドラッグして、Neck 1 Rotate、Neck 2 Rotate、Jaw Rotate と名前を付けます。***spine1***、***spine2***、***spine3*** を **Body** フォルダにドラッグして、***Spine 1***、***Spine 2***、***Spine 3*** と名前を付けます。COG パラメータと背骨パラメータの間に Separator を追加します。

Accept をクリックします。これで新しいコントロールがリグに保存されました。***test_rig*** でコントロールを試してみましょう。

背骨のコントロール

このリグでは、ジョイントの回転を使用して背骨をセットアップしました。これは**フォワードキネマティクス**と呼ばれるものです。リグの他のパーツと同じく、メインのスケルトンのジョイントをベースに、Rig Pose でセットアップして、トップレベルにプロモートします。

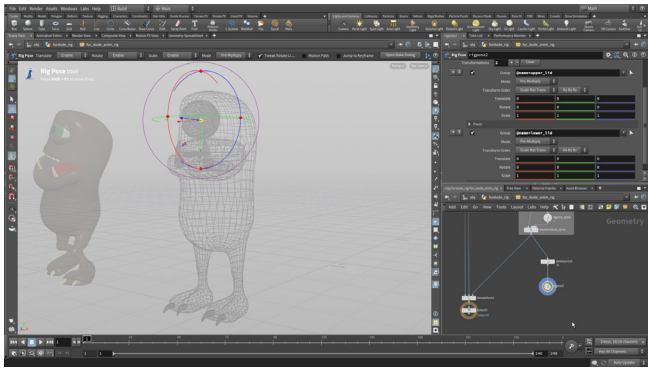
これらのパーツにはコントロールジオメトリは使用しませんでした。Scene View でジョイントを簡単に選択できるからです。すべてのジョイントにコントロールジオメトリが必要なわけではありません。



パート 15

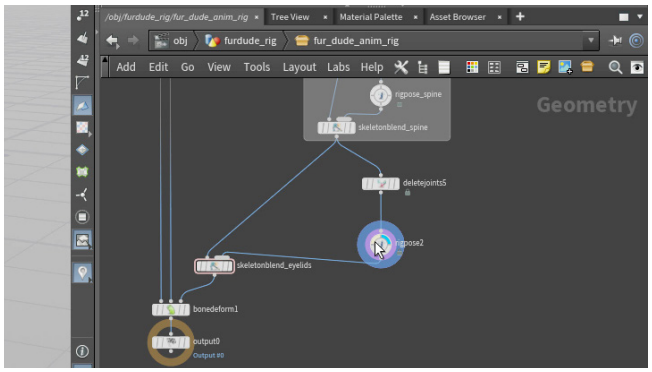
目のコントロール

次は、コントロールジオメトリを使ってまぶた（eyelid）をセットアップし、重なり合ったジョイントを選択しやすくします。また、VOP と呼ばれる Houdini の別のセクションを使用して、目のターゲットジョイントを眼球の Look At としてセットアップします。これらのパーツのリギングが終了したら、適切なパラメータをキャラクタのアセットに再度プロモートします。

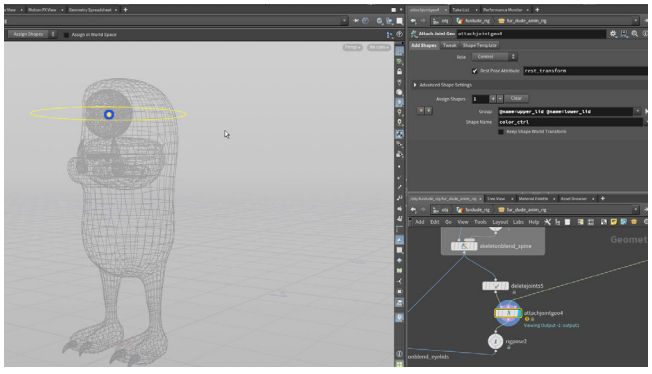


01 ***skeletonblend_spine*** ノードから **Delete Joints** ノードを分岐させて、**Display フラグ**を設定し、***deletejoints_eyelids*** と名前を付けます。**Group** の横の矢印をクリックし、**Rig Tree** で ***upper_lid*** と ***lower_lid*** を選択します。カーソルを Scene View に移動して **Enter** を押したら、**Operation** を **Delete Non-Selected** に設定します。

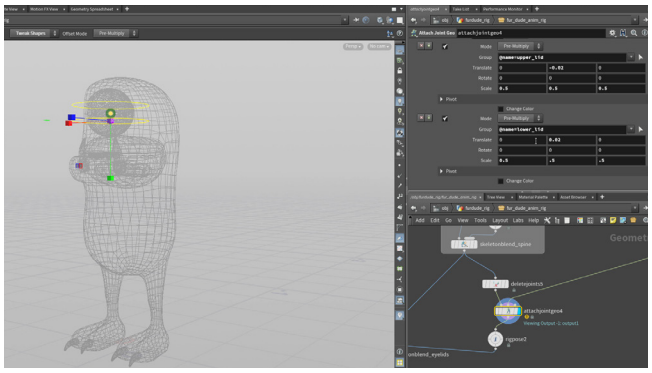
Rig Pose ノードを追加して、名前を ***rigpose_eyelids*** に変更します。**Display フラグ**を設定したら、Scene View で **S** キーを押しながらすべてのジョイントを選択します。これらのジョイントがリグポーズのリストに追加されます。



02 ***skeletonblend_spine*** と ***bonedeform*** ノードの間に **Skeleton Blend** ノードを追加します。このノードの名前を ***skeletonblend_eyelids*** に変更し、**World Space** チェックボックスを **オン**にして、***weight1*** を **1** に設定します。その 2 つ目の入力に ***rigpose_eyelids*** ノードの出力を接続します。



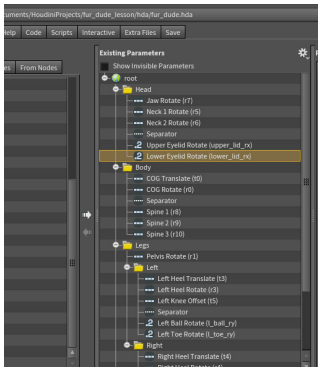
03 ***deletejoints*** ノードと ***rigpose*** ノードの間に **Attach Joint Geometry** ジオメトリを配置します。その 2 つ目の入力に ***mergepacked*** ノードを接続します。このノードに **Display フラグ**を設定し、Scene View で **Enter** を押します。**Mode** が **Assign Shapes** に設定されていることを確認します。まぶたのジョイントを 2 つとも選択したら、**G を押し**、スクロールホイールを使用して ***circle_ctrl*** ジオメトリを見つけます。

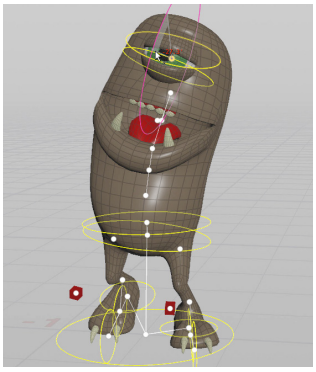


04 上部のオペレーションコントロールツールバーで、**Mode** を **Tweak Shapes** に変更します。

eyelid ジョイントを選択し、**G を押して**トランスフォームハンドルを表示します。**E を押して**スケールハンドルにしたら、中央のハンドルをクリック & ドラッグして、これらのコントロールが少し小さくなるまで 3 方向すべてでスケールします（パラメータエディタで約 0.5）。

upper_eyelid ジョイントを選択して、上に **0.02** 移動します。次に、***lower_eyelid*** ジョイントを選択して、下に **0.02** 移動します。





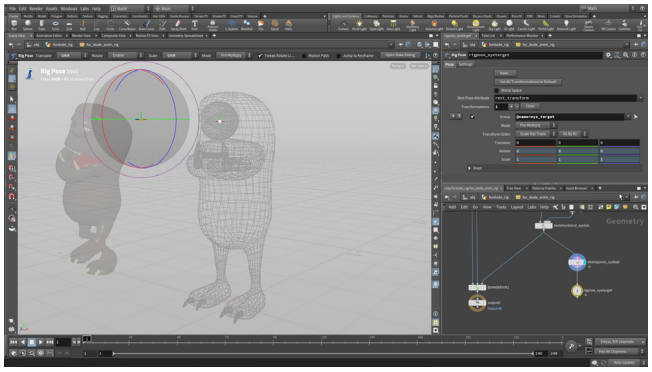
05

Assets メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。**Parameters** タブをクリックします。

これらすべてのジョイントについて、**Translate**、**Rotate**、**Scale** パラメータをロックします。**Rotate X** パラメータをロック解除します。***rigpose_eyelids*** ノードで、***Rotate X*** を ***upper_lid*** から **Head** フォルダにドラッグします。**Range** を **-10, 30** に設定します。次に、***Rotate X*** を ***lower_lid*** から **Head** フォルダにドラッグします。**Range** を **-20, 20** に設定します。

Separator を追加して、目のコントロールを頭部のコントロールから分割します。

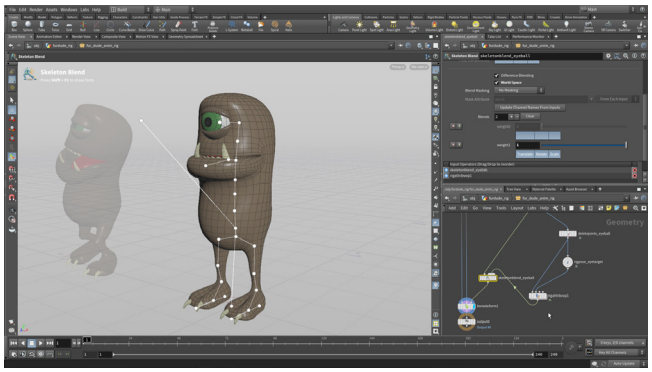
Accept をクリックします。これで新しいコントロールがリグに保存されました。***test_rig*** でコントロールを試してみましょう。



06

skeletonblend_eyelids ノードから **Delete Joints** ノードを分岐させて、**Display フラグ**を設定し、***deletejoints_eyes*** と名前を付けます。**Group** の横の矢印をクリックし、Scene View で ***eyeball*** および ***eye_target*** ジョイントを選択します。**Enter** を押し、**Operation** を **Delete Non-Selected** に設定します。

Rig Pose ノードを追加して、名前を ***rigpose_eyetarget*** に変更します。**Display フラグ**を設定し、Scene View で ***eye_target*** ジョイントを選択します。そのジョイントがリグポーズのリストに追加されます。Look At 拘束でコントロールするので、***eyeball*** ジョイントは必要ありません。***eye_target*** ジョイントについて、**Rotate** および **Scale** パラメータをロックします。

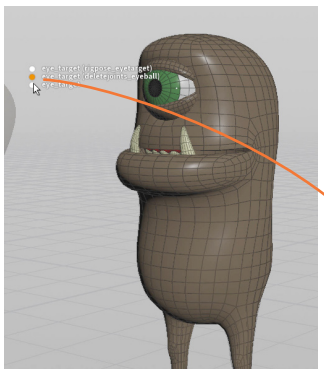


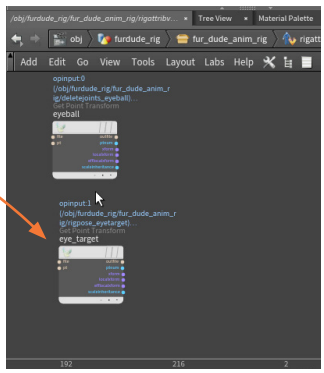
07

Rig Attribute VOP ノードを追加します。1 つ目の入力に ***deletejoints_eyes*** ノードを、2 つ目の入力に ***rigpose_eyetarget*** ノードを接続します。

skeletonblend_eyelids と ***bonedeform*** ノードの間に **Skeleton Blend** ノードを追加します。このノードの名前を ***skeletonblend_eyeball*** に変更し、**World Space** チェックボックスを**オン**にして、***weight1*** を **1** に設定します。その 2 つ目の入力に ***rigattributevop*** ノードの出力を接続します。

bonedeform ノードに **Display フラグ**を設定します。

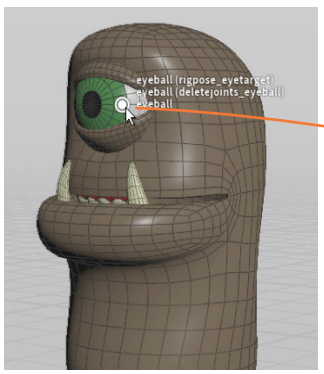


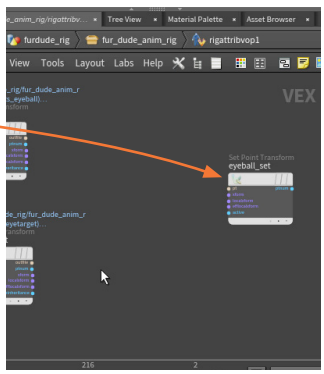


08

rigattributevop ノードをダブルクリックして、中に入ります。Scene View で ***eyeball*** ジョイントをクリックし、***eyeball (deletejoints)*** バージョンをネットワークエディタにドラッグします。これにより **Get Point Transform** ノードが配置されます。これは **First Input** からの ***eyeball*** ジョイントにフォーカスするノードです。

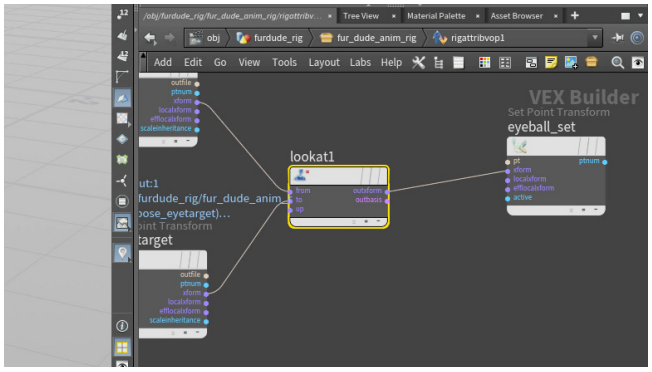
eye_target ジョイントをクリックし、***eyetarget (ripose_eyetarget)*** バージョンをネットワークエディタにドラッグします。これにより **Get Point Transform** ノードが配置されます。これは **Second Input** からの ***eye-target*** ジョイントにフォーカスするノードです。





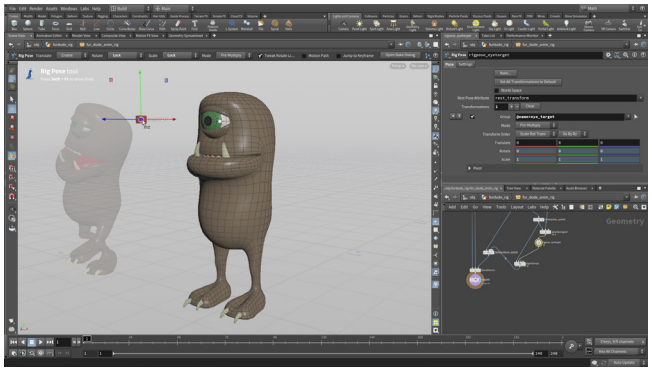
09

eyeball ジョイントをクリックし、***eyeball*** バージョンをネットワークエディタにドラッグします。**Set Point Transform** ノードが配置されます。これは ***eyeball*** ジョイントにフォーカスするノードです。



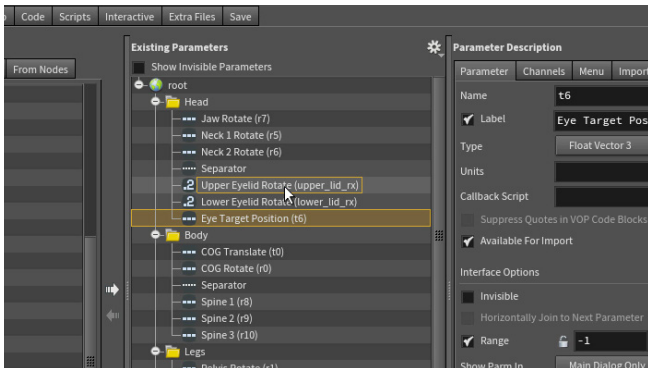
10 **Tab > Look At (KineFX)** を選択し、ノードを中央に配置します。eyeball ***getpointtransform*** ノードの **xform** 出力を、***lookat*** ノードの **from** 入力に接続します。eye_target ***getpointtransform*** ノードの **xform** 出力を、***lookat*** ノードの **to** 入力に接続します。***lookat*** ノードの **outxform** 出力を、***eyeball_set*** ノードの **xform** 入力に接続します。

眼球のジオメトリが反転しています。***lookat*** ノードを選択し、**Look At Axis** を **Z** に設定すると、最初にリグをセットアップしたときに設定した方向と一致します。



11 **Attach Control Geometry** ノードを ***deletejoints_eyes*** ノードと ***rigpose_eyetarget*** ノードの間に配置します。その 2 つ目の入力に、先ほどのネットワークから ***mergepacked*** ノードを接続します。このノードに Display フラグを設定して、Handle ツールに移動します。**Mode** が **Assign Shapes** に設定されていることを確認します。3D ビューで ***eye_target*** ジョイントを選択したら、**G を押し**、スクロールホイールを使用して ***square_ctrl*** ジオメトリを見つけます。

bonedeform ノードに **Display フラグ**を設定します。***rigpose_target*** をクリックします。***eye_target*** ジョイントを選択して動かすと、それに合わせて眼球の向きが変わります。**Undo** を実行して元の位置に戻します。

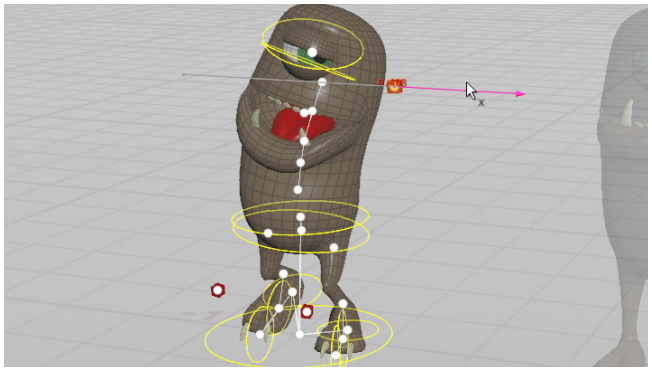


12 目を定義するのに使用したすべてのノードを囲む**ネットワークボックス**を追加し、***Eye Controls*** と名前を付けます。

Assets メニューから、**Edit Asset Properties > Fur Dude Anim Rig** を選択します。**Parameters** タブをクリックします。

rigpose_eyetarget ノードで、***Translate パラメータ***を ***eye_target*** から **Head** フォルダの***目 (eye)*** のセクションにドラッグします。**Eye Target Position** と名前を付けます。

Accept をクリックします。これで新しいコントロールがリグに保存されました。



13 ***test_rig*** でコントロールを試してみましょう。

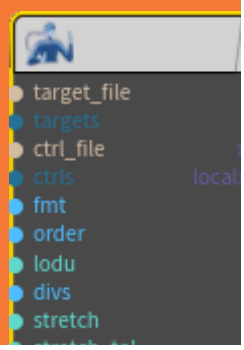
このコントロールリグのすべてのパーツが完成しました。歩行サイクルのアニメーションを付け始められます。テストリグのコピーをもう 1 つ作成し、そのネットワークを使ってアニメートします。

このデジタルアセットは、複数のシーンファイルに複数のインスタンスを作成できます。後で戻って変更を加えた場合、すべてのアセットが更新されます。これが、デジタルアセットリグを使用したパイプラインの利点です。

VOP でのリギング

Rig Attribute VOP は、このレッスンで紹介したもの以外にもさまざまなソリューションを提供します。IK チェーンはこの方法で構築でき、前に使用した IK チェーン SOP にはその 1 つが含まれています。

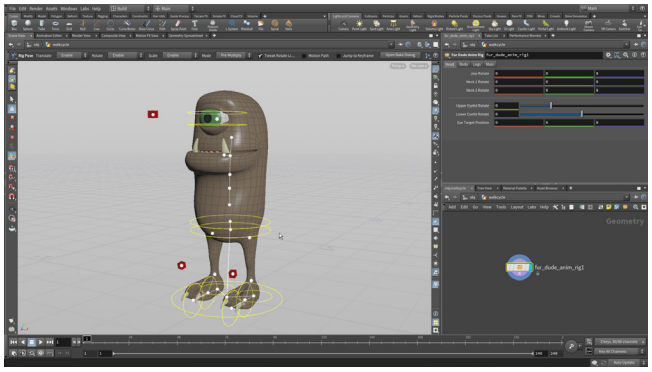
VOP を使用すると、Curve Solver、Realistic Shoulder、リバースフットなどもセットアップできます。Scene View から VOP ネットワークにジョイントをドラッグできるこれまでにないワークフローは、ワークフローを高速化します。



パート 16

リグのアニメーション

ファー・デュード(Fur Dude)にキーフレームを設定して、歩行サイクルを作成します。Channel List を使用してチャンネルをピン留めし、動きをブロッキングするなど、新しいツールも使用します。簡単な歩行サイクルを完成させて、動くファー・デュードを確認しましょう。ここでの目標は、基本のキーフレームワークフローを計画し、KineFX リグのアニメーション方法を理解することです。

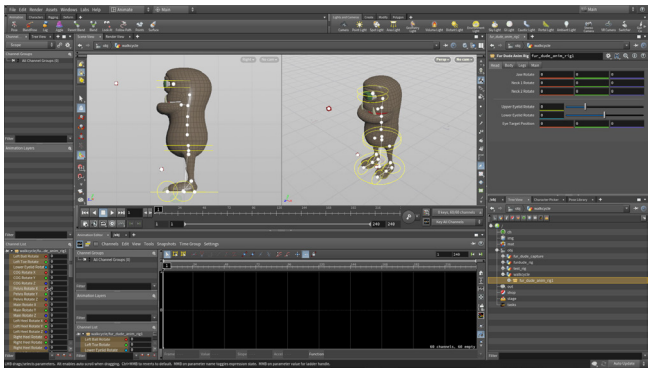


01

オブジェクトレベルに移動します。**Tab > Geometry** を選択してノードを配置します。名前を walkcycle に変更します。すべてのオブジェクトの **Display フラグ**をオフにします。

ダブルクリックして ***walkcycle*** の中に入り、ネットワークビューで **Tab > Fur Dude Anim Rig** を選択します。**Enter** を押して、原点に配置します。これは新しいロックバージョンの Fur Dude リグで、ゼロからアニメートしていきます。

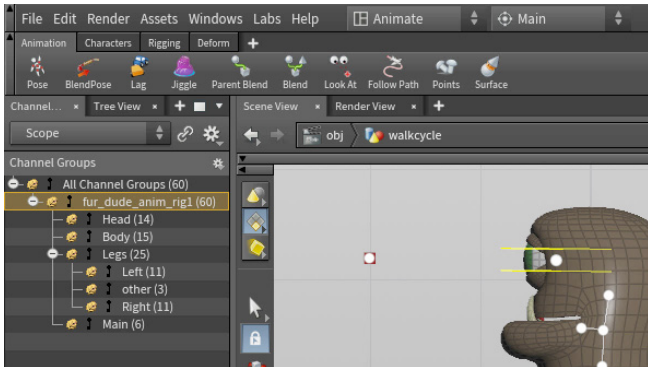
この方法では、シーンに別バージョンのキャラクタアセットを配置します。このシーンファイル(または任意のシーンファイル)には、複数のバージョンを配置でき、いずれもディスク上の同じアセット定義を参照します。



02

Desktop メニュー(ここでは Build)から、**Animate** を選択します。キーフレームワークフローに対応したパネルが表示されます。場合によっては、***walkcycle*** オブジェクトに戻る必要があります。

左側の **Channel List** は、キャラクタのアニメーションのブロッキングで重要な役割を果たします。**アニメーションエディタ**では、アニメーションカーブを表示および編集できます。このレッスンでは、動きをブロッキングするまでにとどめ、カーブの編集は行いません。



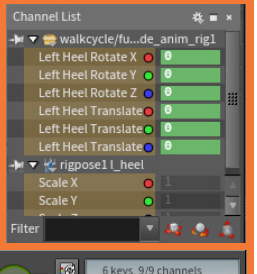
03

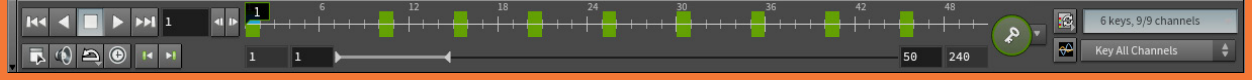
ネットワークビューで、***walkcycle*** オブジェクト内の ***fur_dude_anim_rig*** を選択します。パラメータエディタで、右上のボックスアイコンをクリックします。**Parameters and Channels > Create Nested Channel Groups** を選択します。ポップアップウィンドウで **Close** をクリックします。アセットのパラメータがリストされ、フォルダ別に整理されています。

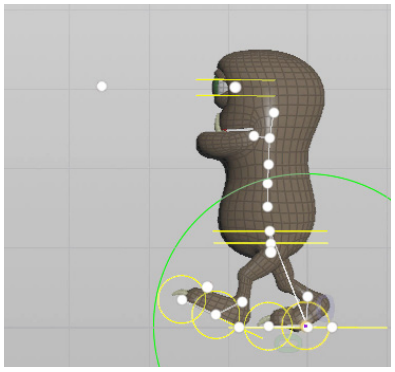
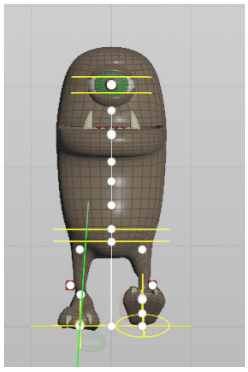
fur_dude_anim_rig チャンネルグループの横の**ピン**アイコンをクリックして、これらのチャンネルをピン留めします。タイムラインが**フレーム 1** に設定されていることを確認したら、**K を押して**、すべてのチャンネルにキーフレームを設定します。

チャンネルの動作

ジョイントを選択すると、そのチャンネルがチャンネルリストに読み込まれます。**K** を押すと、該当のジョイントにキーフレームを設定できます。ブロッキングのためにチャンネルを読み込んだままにしたい場合は、チャンネルをピン留めするか、チャンネルグループとしてまとめてピン留めします。チャンネルグループはアセットから直接作成できます。グループは、キャラクタ用の UI をどのように構築したかに基づいて編成されます。独自のグループを構築して、特定のチャンネルをピン留めすることも可能です。

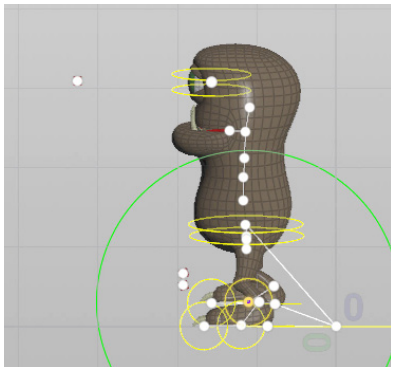
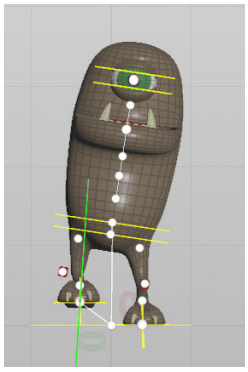




04 **タイムライン**を **10** で開始し、**50** で終了するように設定します。**フレーム 10** に移動します。すべてのチャンネルをピン留めした状態のまま、**K を押し**、もう 1 つキーフレームを設定します。**先にキーフレーム**を設定してから、ポーズを付けることをお勧めします。ポージングによって、キーフレームを設定したフレームで値が更新されます。

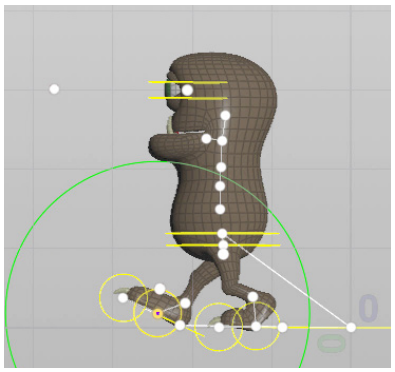
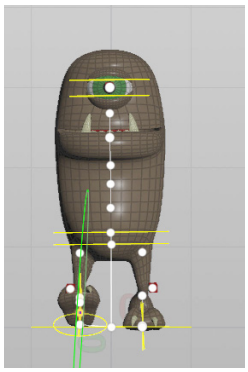
左の heel を前に動かし、上に回転させたポーズを作成します。***COG*** を少し下げ、***右の ball*** を前方に回転させます。

05 **フレーム 15** に移動し、**K を押します**。

左のかかとを回転して平らにします。ただし、移動はしません。左足の動きに合わせて COG を動かします。左足に合わせて、***右の heel*** を上に移動します。***右の ball*** を回転して元に戻し、足を平らにします。

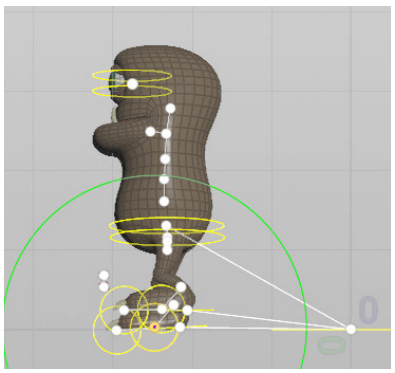
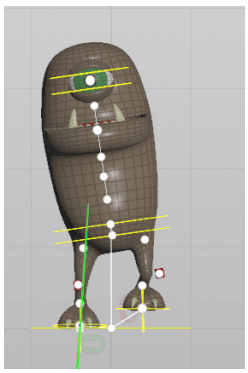
COG を左足の方に少し回転させます。***背骨***の 3 つのジョイントをいくらか回転させて、この傾きを強調してもよいでしょう。

06 **フレーム 20** に移動し、**K を押します**。

右の heel を前に動かし、上に回転させたポーズを作成します。***COG*** を少し下げ、***左の ball*** を前方に回転させます。つまり、フレーム 10 とは逆のポーズです。

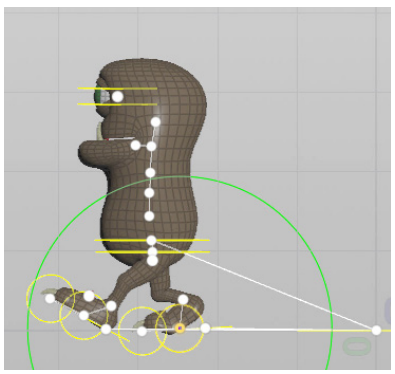
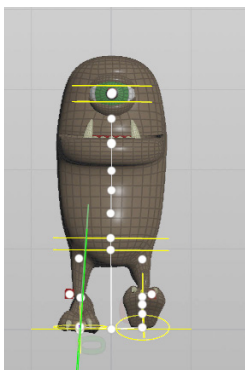
COG と***背骨***のジョイントを回転して中央に戻します。

07 **フレーム 25** に移動し、**K を押します**。

右の heel を回転して平らにします。ただし、移動はしません。右足の動きに合わせて ***COG*** を動かします。左足に合わせて、***左の heel*** を上に移動します。***左の ball*** を回転して元に戻し、足を平らにします。

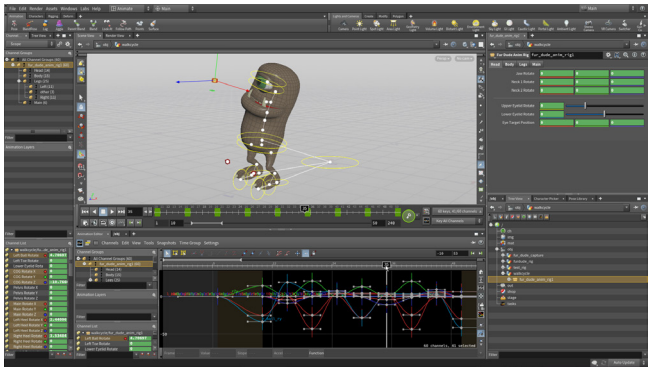
COG を右足の方に少し回転させます。***背骨***の 3 つのジョイントをいくらか回転させて、この傾きを強調してもよいでしょう。

08 **フレーム 30** に移動し、**K を押します**。

左の heel を前に動かし、上に回転させたポーズを作成します。***COG*** を少し下げ、***右の ball*** を前方に回転させます。

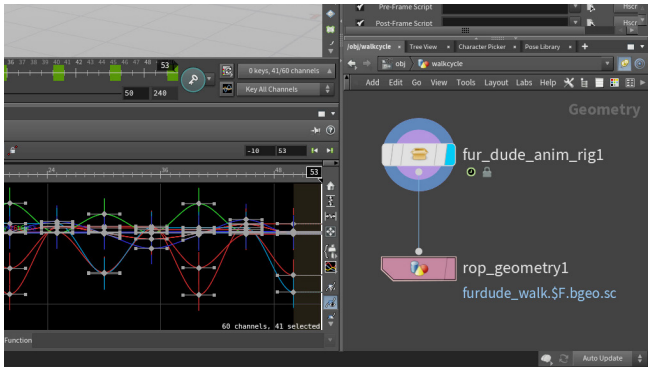
COG と 背骨 のジョイントを回転して中央に戻します。



09

このパターンをフレーム **50** まで続けます。同じポーズを繰り返すと、前に進む歩行サイクルを作成できます。

この時点で、戻ってポーズを調整したり、動きを整えることができます。キーフレームをさらに追加して、追従するアクションを作成してもよいでしょう。目の動きやまばたきをアニメートしたり、もちろん、50 フレーム以上の長いアニメーションにしてもかまいません。



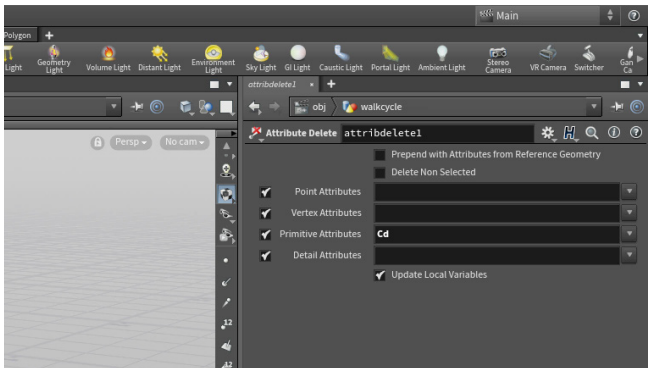
10

fur_dude_anim_rig ノードの出力を ROP Geometry ノードに接続します。これで、ファー・デュードのジオメトリのキャッシュをエクスポートできるようになります。**Output File** を次のように設定します。

```
$HIP/geo/furdude_walk.$F.bgeo.sc
```

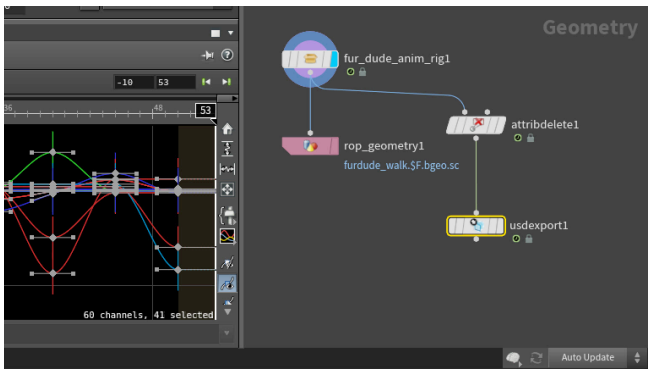
次に、**Valid Frame Range** を **Render Frame Range** に設定します。**Start/End/Inc** パラメータで **RMB クリック**し、**Delete Channels** を選択します。**Start** を **1**、**End** を **50** に設定します。

Save to Disk ボタンをクリックして、キャッシュをディスクに保存します。これを使って後ほどファーを追加します。



11

fur_dude_anim_rig ノードから **Attribute Delete** ノードを分岐させます。**Primitive Attributes** で **Cd** を選択します。これで、全ボディパーツから、カラーが取り除かれます。



12

attributedelete ノードの出力を **USD Export** ノードに接続します。これで、ファー・デュードのジオメトリを USD フォーマットでエクスポートできるようになります。**Output File** を次のように設定します。

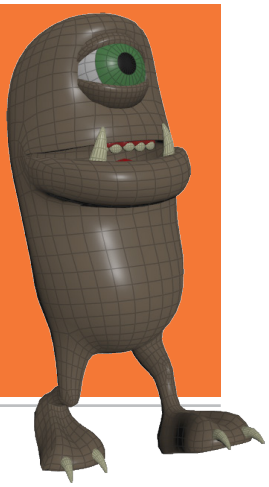
```
$HIP/usd/furdude_walk.usd
```

次に、**Valid Frame Range** を **Render Frame Range** に設定します。**Start/End/Inc** パラメータで **RMB クリック**し、**Delete Channels** を選択します。**Start** を **1**、**End** を **50** に設定します。

Save to Disk ボタンをクリックして、キャッシュを USD ファイルに保存します。このファイルは、後のレンダリングプロセスで使用します。

アニメーションのキャッシュ化

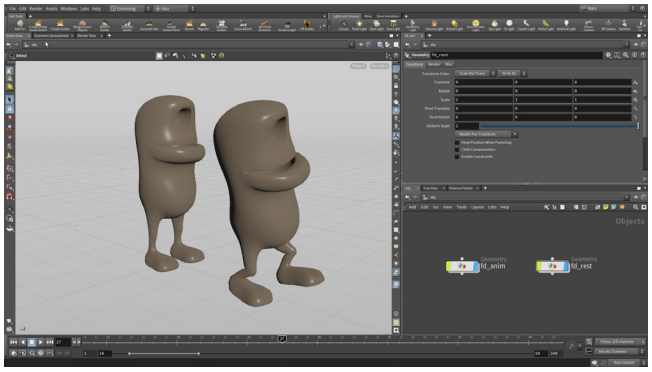
Houdini のワークフローはプロシージャルなため、アニメーションのキャッシュ化は絶対に必要なわけではありません。このネットワークのアニメーションは、任意のネットワークで参照してグルーミングすることもできますし、Solaris で参照するために USD に変換することも可能です。キャッシュ化のメリットは、アニメーションを固定したり、統合したディスク上のファイルを使用できることです。これは、プロダクションにとって非常に好都合なアプローチです。Solaris では、ディスクを参照する USD ファイルの方が効率的でもあります。Houdini はディスクからファイルを参照するので、いつでも自由にアニメーションを変更したり、新しいシーケンスを出力できます。また、それらを自動的に取得するようにできます。



パート 17

ファーの追加とグルーム

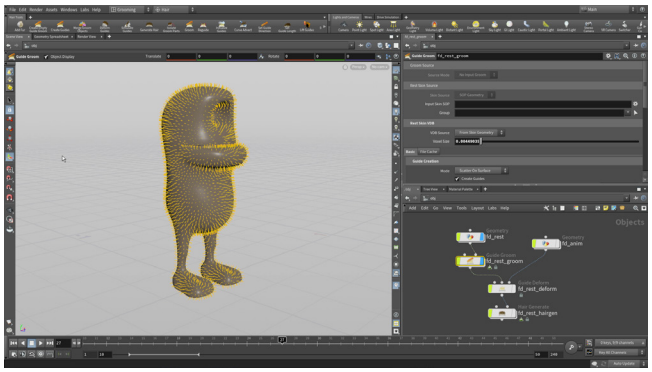
このキャラクタをファー・デュード（Fur Dude）と名付けたのには理由があります。さまざまなグルーミングツールを使用して、もじゃもじゃヘアーを追加し、形状を整えていこうというわけです。グルーミング用に設計されたデスクトップを使用して、縮毛、クランプ（束）、ヘアーのダイナミクスを追加し、Fur Dude の歩行に合わせてシミュレートします。完成したら、レンダリング用にエクスポートします。



01

Grooming デスクトップに切り替えます。4 つの既存のオブジェクトをネットワークボックスで囲み、そのボックスに ***Rig & Animate*** と名前を付けます。

ネットワークエディタで、**Tab > File** を選択します。ノードを配置したら、ダブルクリックして中に入ります。**Geometry File** の横の**ブラウズ**ボタンをクリックして、***$HIP/geo*** に移動します。***furdude_walk.$F.bgeo.sc*** を選択し、**Accept** を押します。**Blast** ノードを追加し、**Group** を ***fur_dude_body*** に設定します。体に集中できるよう、**Delete Non-Selected** チェックボックスを**オン**にします。**Display フラグ**を設定したら、オブジェクトレベルに移動して、名前を ***fd_anim*** に変更します。そのオブジェクトを **Alt ドラッグ**してコピーを作成し、fd_rest と名前を付けます。その中に入り、**Geometry File** を ***$HIP/geo/ furdude_walk.1.bgeo.sc*** に変更します。

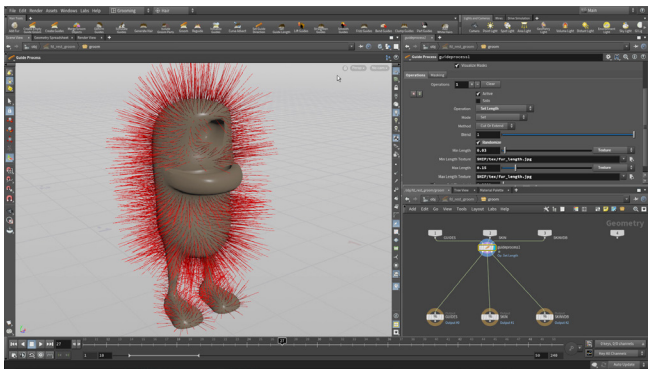


02

タイムスライダを少し進めましょう。片方のオブジェクトはファー・デュードの静的バージョンで、もう一方はアニメートされています。**Add Fur** ボタンをクリックします。***fd_rest*** ジョイントを選択して、**Enter** を押します。

次は ***fd_anim*** ジョイントを選択して、**Enter** を押します。***fd_rest_anim***、***fd_rest_deform***、***fd_rest_hairgen*** の **Display フラグ**をオフにします。

fd_rest および ***fd_rest_groom*** の **Display フラグ**をオンにします。



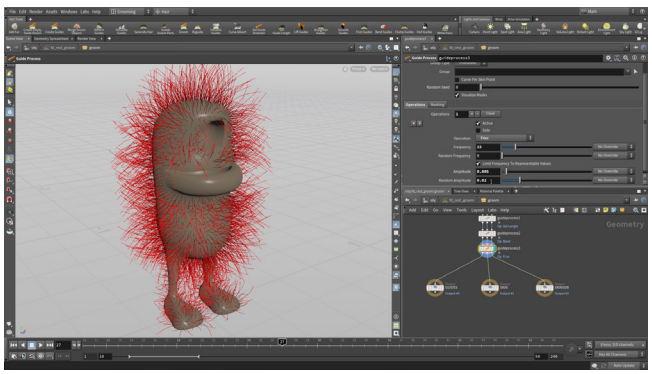
03

fd_rest_groom ノードを選択し、**Hair Tools** シェルフで **Set Guide Length** をクリックします。Randomize ボタンをオンにします。**Min Length** を **0.03** に設定し、右側のメニューから **Texture** を選択します。

ファイルブラウザボタンを使用して ***$HIP*** を選択したら、***tex*** ディレクトリに移動して ***fur_length.jpg*** を選択します。

Max Length を **0.15** に設定し、**Texture** を再度選択します。右側の**矢印**を使用して、***fur_length.jpg*** を選択します。

これで、目、唇、足の裏がマスクアウトされ、残ったファーの長さはランダムになります。



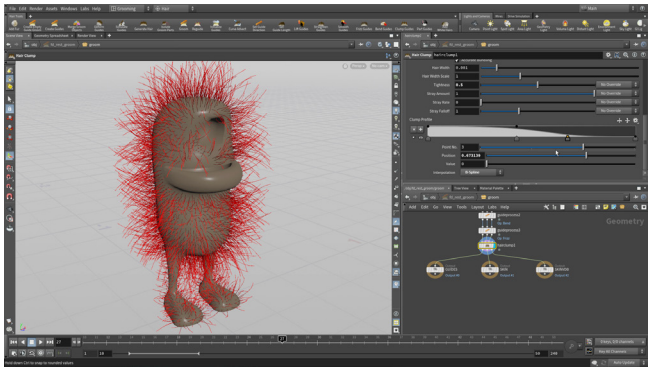
04

Hair Tools シェルフで **Bend Guides** をクリックします。**Angle** を **45** に設定して、ガイドをいくらか曲げます。

次は **Frizz Guides** ツールをクリックします。次のように設定します。

- **Frequency** を **15** にする
- **Amplitude** を **0.005** にする
- **Random Amplitude** を **0.02** にする

これで、ヘアーをレンダリングしたとき、真っすぐすぎません。モジャモジャ感を増したい場合は、縮毛を追加します。



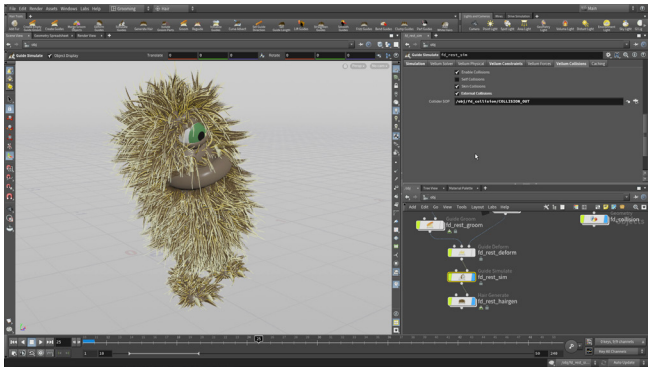
05

Hair Tools シェルフで **Clump Guides** をクリックします。**Clump Size** を **0.02**、**Tightness** を **0.5** に設定します。

Clump Profile を変更します。束の根本から中間では高い値を維持し、ヘアーの先端付近で徐々に細めます。

オブジェクトレベルに移動し、**fd_rest** と ***fd_rest_groom*** の Display フラグを**オフ**にします。そして、***fd_anim***、***fd_rest_sim***、***fd_rest_hairgen*** の Display フラグを**オン**にします。

fd_rest_groom ノードを選択し、**Density** を **20000** に設定します。

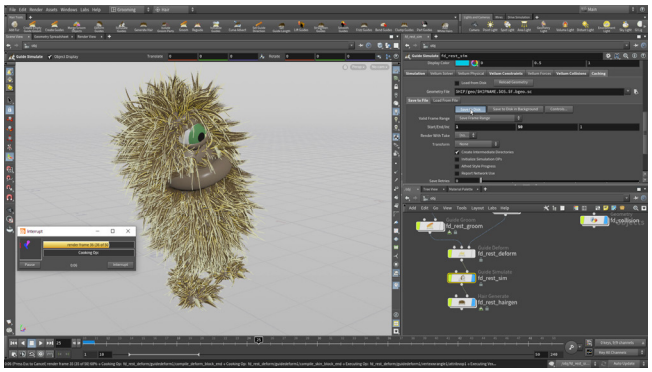


06

fd_rest deform ノードを選択し、**Hair Tools** シェルフで **Simulate Guides** をクリックします。***fd_rest_sim*** ノードで、**Vellum Constraints** タブに移動し、**Bend** の **Stiffness** を **5** に設定します。

fd_anim ノードを **Alt ドラッグ**してコピーを作成し、***fd_collision*** と名前を付けます。このノードの中に入り、***blast*** ノードで **Delete Non Selected** を**オフ**にします。**Group** に舌、上の歯、歯茎を追加したら、Null ノードを追加して ***COLLISION_OUT*** と名前を付けます。

オブジェクトレベルで、fd_rest_sim を選択します。**Vellum Collisions** で **External Collisons** を**オン**にして、**Collider SOP** を ***../fd_collision/COLLISION_OUT*** に設定します。



07

Caching タブをクリックして、**Valid Frame Range** を **Save Frame Range** に設定します。**Start/End/Inc** パラメータで **RMB クリック**し、**Delete Channels** を選択します。**Start** を **1**、**End** を **50** に設定します。**Save to Disk** をクリックして、シミュレーションを実行します。

ここで **Load from Disk** チェックボックスを**オン**にします。フレーム毎にヘアーを計算するのではなく、キャッシュを使用してファーが定義されるようになります。

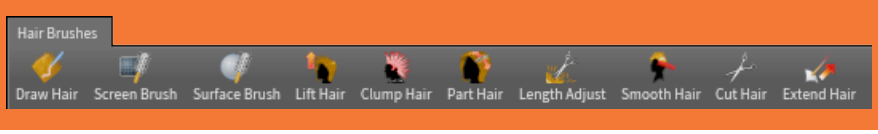
08

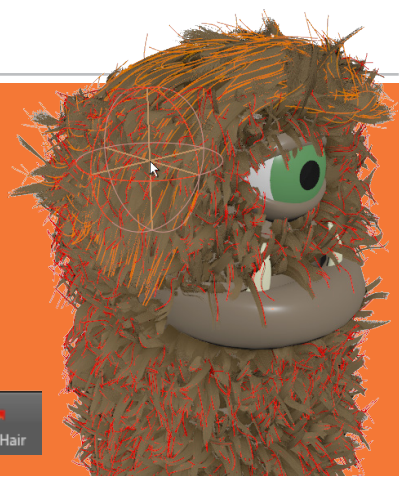
fd_rest_hairgen ノードを選択します。**Distribution** の **Density** を **1000000** に設定します。Guide Interpolation にスクロールし、**Clump Crossover** を **0.25** に設定して、束同士が少し重なるようにします。これで、ふさふさのヘアーを持つファー・デュードの見た目が定まります。

これらは、Solaris と呼ばれる Houdini のライティングコンテキストでレンダリングするヘアーではありません。代わりに、ガイドヘアーを取り込んで、レンダリング時にヘアープロシージャルを使用してレンダリングします。

ヘアーブラシ

グルーミングデスクトップには、キャラクタのサーフェス上でインタラクティブに使用できるヘアーブラシツールもあります。ヘアーを長くしたり、滑らかにしたり、カットしたり、伸ばすことができます。ファー・デュードのグルーミングには使いませんでしたが、後でこれらを使用して、最終的なルックに磨きをかけてもよいでしょう。





パート 18

ショットの設定とレンダリング

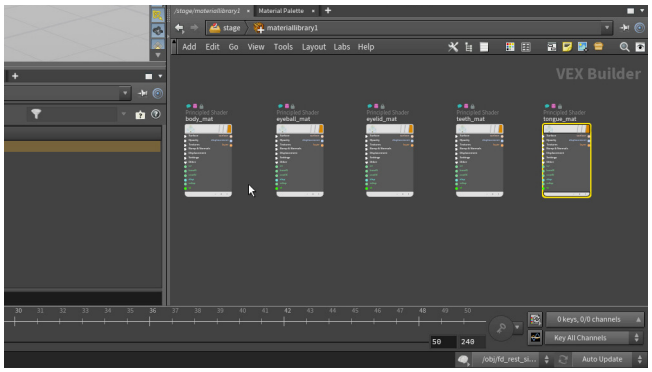
ショットのレンダリング時には、USD ファイルを Solaris ステージで参照し、背景を追加します。Solaris は Houdini のコンテキストで、LOP ノードを使用して USD シーングラフをセットアップします。次に、ファーを読み込んで、カメラとライトを配置します。その後、Karma レンダラを使ってショットのプレビューレンダリングを作成してから、アニメーションシーケンスをレンダリングします。

01

デスクトップを **Solaris** に変更します。パスバーで **Stage** を選択します。ネットワークビューで **Tab > Reference** を選択してからクリックし、**Reference** ノードを追加します。

Reference File の横にある **File Pattern** をクリックして、***furdude_walk.usd*** を指定します。ノードの名前を ***furdude*** に変更します。**Primitive Path** を ***/char/`@sourcename`*** に設定し、ノード名を使用して、***char*** というグループに配置されるようにします。**シーングラフツリー**で ***char***、***furdude*** の順に展開すると、名前の付いたプリミティブをすべて確認できます。

Scene View で、ビューをホームする**スペースバー + H** のような表示ツールを使用して、歩行サイクルがよく見えるようにします。

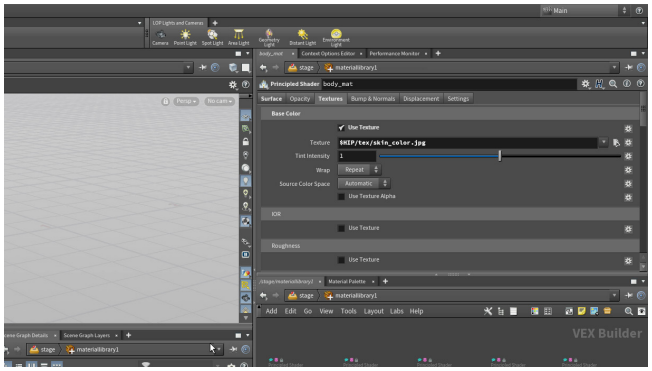


02

Tab > Material Library を選択します。それを ***reference*** ノードの出力に接続し、**Display フラグ**を設定します。

Material Palette ペインに移動します。***/stage/materiallibrary*** の横にある矢印をクリックし、このエリアを開きます。パレットの左側のマテリアルギャラリーをスクロールして、**Principled Shader** マテリアルを ***materiallibrary*** 作業エリアにドラッグします。

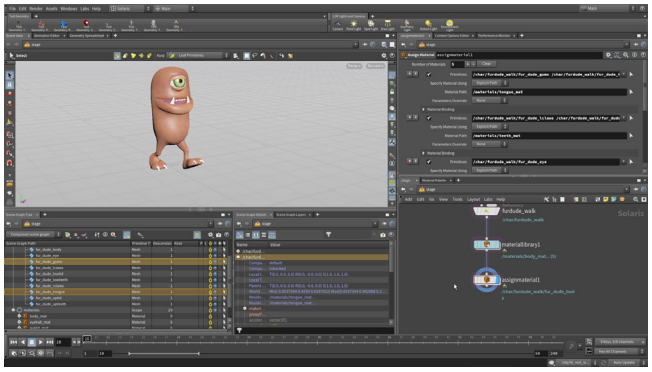
ネットワークビューに移動して、このマテリアルを **Alt ドラッグ**してさらに 4 つ作成します。5 つのマテリアルの名前を、それぞれ ***body_mat***、***eyeball_mat***、***eyelid_mat***、***teeth_mat***、***tongue_mat*** に変更します。マテリアルは、**シーングラフツリー**でも確認できます。



03

furdude_body_mat で、**Surface** タブの **Base Color** を **1, 1, 1** に設定します。**Textures** タブをクリックし、**Base Color** の **Use Texture** をクリックしてオンにしたら、**Texture** の横のボタンを使用してファイルウィンドウを開きます。左側のリストで **$HIP** をクリックしたら、***tex*** フォルダをクリックして開き、***skin_color.jpg*** をワンクリックして選択します。**Accept** をクリックし、テクスチャをマテリアルに割り当てます。次に、**Roughness** を **0.5**、**Reflectivity** を **0** に設定します。

同じ手順で、***eye_color.jpg*** と ***eye_lid.jpg*** をそれぞれのマテリアルに割り当てます。***tongue_mat*** を**赤に近いピンク**に、***teeth_mat*** を**黄色がかった白**に設定します。



04

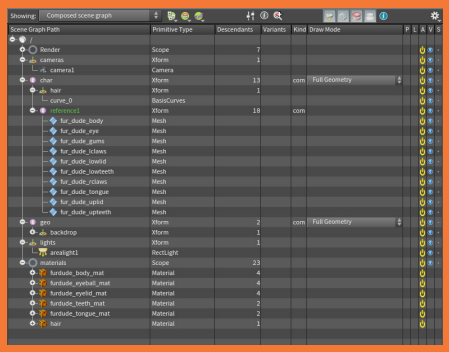
ステージレベルに戻ります。***Material Library*** の後に **Assign Material** ノードを追加します。シーングラフから ***fur_dude_body*** を **Primitives** フィールドにドラッグしたら、**Material Path** の横にある矢印をクリックして、***body_mat*** を選択します。横にある **+(プラス)記号**をクリックして、4 つの新しいエントリを追加します。次のように割り当てます。

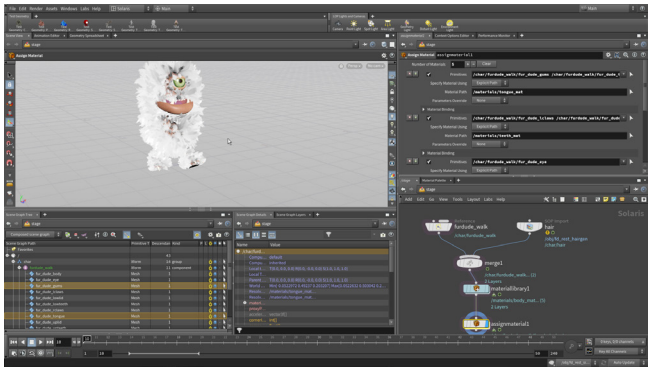
- ***fur_dude_eye*** > ***eyeball_mat***
- ***fur_dude_lowid/uplid*** > ***eyelid_mat***
- ***fur_dude_lowteeth/upteeth/claws*** > ***teeth_mat***
- ***fur_dude_tongue/gums*** > ***tongue_mat***

USD シーングラフ

Solaris で作業するとき、LOP ノードを使用して追加するジオメトリやマテリアルは、**シーングラフ**に追加され、USD に変換されます。ライトやカメラを追加すると、それらも USD シーングラフの一部になります。

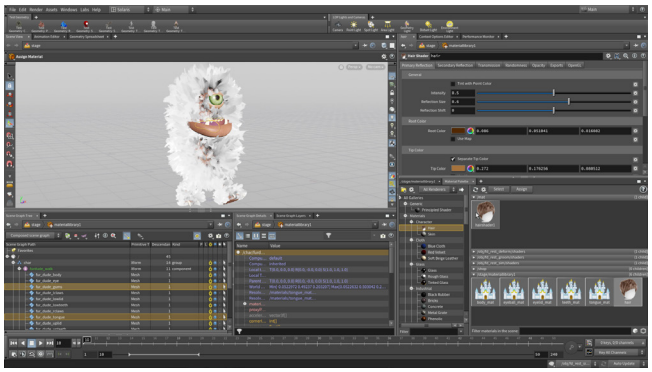
アーティストの皆さんは、Houdini でライティングやレンダリングを行ううえで、USD を完璧に理解する必要はありません。しかし、プロジェクトのパイプラインの観点から考えると、USD はショット管理に便利なツールとなるはずです。





05 ネットワークビューで、**Tab** を押して **SOP Import** と入力します。クリックしてノードを配置します。***hair*** と名前を変更します。**Import Path Prefix** を ***/char/$OS*** に設定します。SOP Path の横のノードアイコンをクリックして、***fd_rest_hairgen*** ノードに移動します。

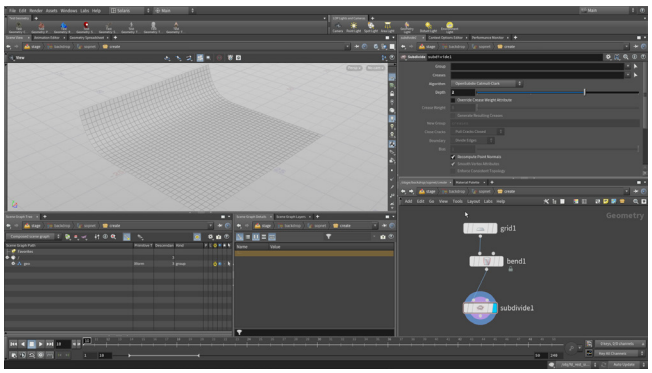
Merge ノードを ***furdude*** と ***materiallibrary*** ノードの間に追加します。***hair*** ノードをそれに接続します。



06 **Material Palette** ペインに移動します。***/stage/materiallibrary*** を開きます。**Hair** マテリアルを ***materiallibrary*** 作業エリアにドラッグします。

ヘアーの **Root Color** と **Tip Color** の設定はデフォルトのままにします。次に **Secondary Reflection** タブをクリックして、**Root Color** を**ダークグレー**、**Tip Color** を**ミディアムグレー**に設定します。

Assign Material ノードに戻り、**Primitives** の横にある矢印をクリックして、**シーングラフ**でファーのカーブを選択します。**Material Path** の横の矢印をクリックして、***hair*** を選択します。

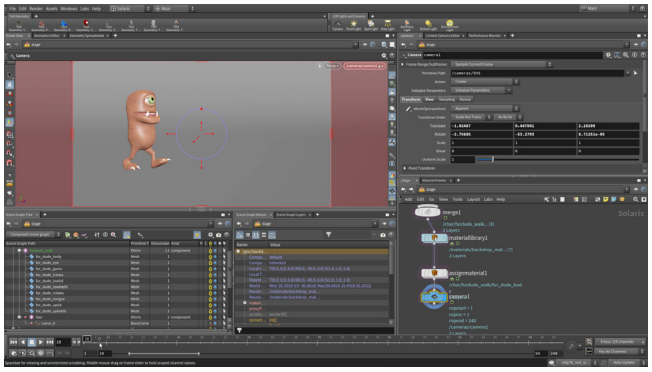


07 ネットワークビューで、**Tab** を押して **Grid** と入力します。クリックしてノードを配置します。ノード名を ***backdrop*** に変更し、***merge*** ノードに接続します。**Import Path Prefix** を ***/geo/$OS*** に設定します。***backdrop*** ノードを**ダブルクリック**して、ジオメトリレベルに入ります。

Grid ノードを選択し、**Size** を **50, 50**、**Rows** および **Columns** を **10** に設定します。***Grid*** ノードの出力を **RMB クリック**して、**Bend** と入力します。クリックして Bend ノードを配置したら、**Display フラグ**を設定します。**Bend** を **75**、**Capture Origin** を **0, 0, -10**、**Capture Direction** を **0, 0, -1**、**Capture Length** を **10** に設定します。***bend*** ノードの出力を **RMB クリック**して、**Subdivide** と入力します。**Display フラグ**を設定し、**Depth** を **2** に設定します。

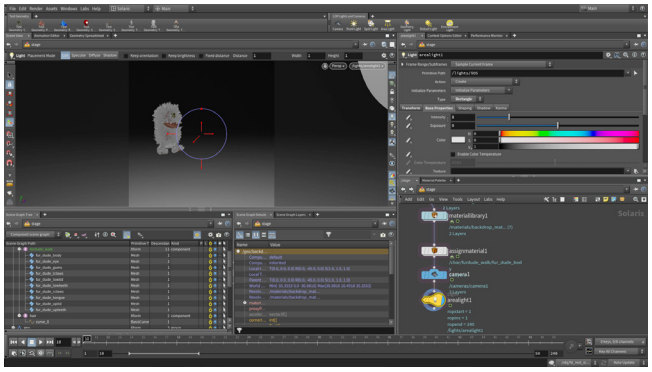
08 オブジェクトレベルに移動し、**Rotate Y** を **-45** 度に設定します。マテリアルを追加して割り当てます。デフォルトのグレーのままにしても、独自のベースカラーを追加してもかまいません。

時間範囲を **10 から 50** に設定します。ヘアーが安定した挙動になるまで 10 フレームほどかかるため、このシーケンスはフレーム 10 からレンダリングします。



09 表示ツールを使用して、正面から ***furdude*** が見えるようにします。**LOP Lights and Camera** シェルフで、**Camera** ツールを **Ctrl クリック**します。ネットワークに camera ノードが加わり、カメラ越しにシーンビューを見られるようになります。

Lock Camera/Light to View ボタンを押し、ビュー変更に応じてカメラの位置が更新されるようにします。Scene View で**タンブル**、**パン**、**ドリー**してカメラを再配置し、ファー・デュードが左から右に動くようにします。タイムラインをスクラブし、シーケンス全体でカメラが機能していることを確認します。



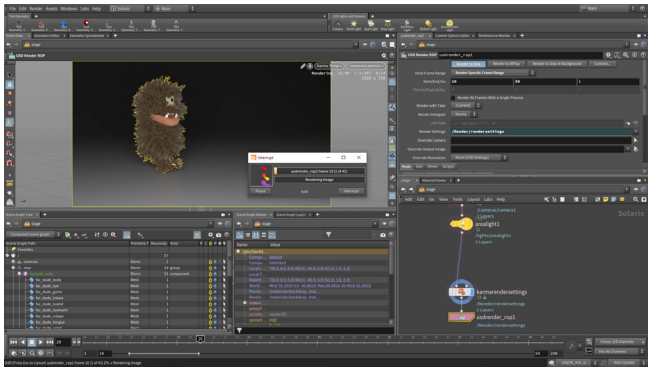
10 **Lock Camera/Light to View** ボタンを**オフ**にしたら、タンブルしてファー・デュードを見下ろすようにしましょう。**LOP Lights and Camera** シェルフで、**Area Light** ツールを **Ctrl クリック**します。***arealight*** ノードをチェーンの終端に追加します。

arealight ノードを選択し、**Base Properties** タブの **Intensity** を 2 に設定します。

11 Persp メニューで **Karma** を選択し、Scene View で Karma を使ってレンダリングします。タイムラインの別フレームに移動すると、Scene View が素早く更新されます。

Karma は USD を使用するよう設計されているので、LOP コンテキストのすべてが USD シーングラフに変換されます。Houdini のこの部分からのみ Karma レンダラを使用できます。

レンダリン時によりクリーンな画像を得るには、Nvidia グラフィックカードを使用している場合には、Denoiser をオンにします。**Render** メニューから **Denoiser** をインストールし、**Display Options** バーでそれをオンにしてください。



12 **Tab > Karma** を押し、**Karma Render Settings** と **USD Render ROP** ノードを追加します。それらをチェーンの終端に接続します。***karmarendersettings*** を選択して、**Image Output > Filters** タブで **Denoiser** を **nvidia Optix Denoiser** に設定します。**Output Picture** を ***$HIP/render/walk/furdude_walk_$F2.exr*** に設定します。名前の ***$F*** は、レンダリングにフレーム番号を付加するためのもので、***2*** はフレーム番号のパディングです。

usdrender_rop を選択します。**Start/End/Inc** パラメータを **RMB クリック**し、**Delete Channels** を選択します。**Start** を **10**、**End** を **50** に設定します。***usdrender_rop*** ノードを選択します。**Render to Disk** をクリックします。

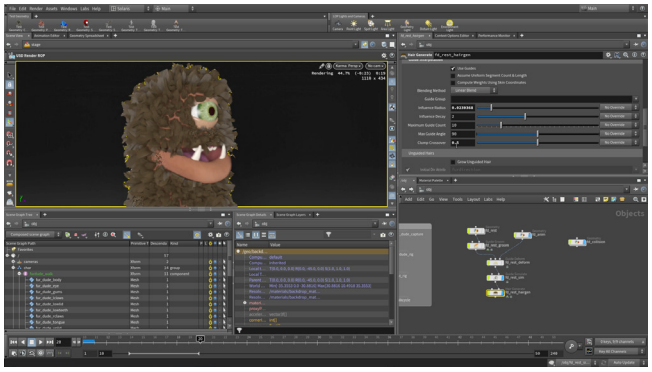
KARMA レンダラ

Houdini のレンダラである Karma を使用して、シーケンスをレンダリングします。Karma は USD をレンダリングするよう設計されており、これはレンダーデリゲートと呼ばれています。最初は Scene View でレンダリングします。Scene View で **D** を押して表示オプションを表示し、レンダリングをコントロールします。Denoiser をオンにしたり、Pixel Samples を設定したり、Image Resolution を定義することができます。

その後 **Karma LOP** をセットアップすると、そのノードのレンダリング設定を使用して最終的な出力を作成し、ディスクに保存できます。

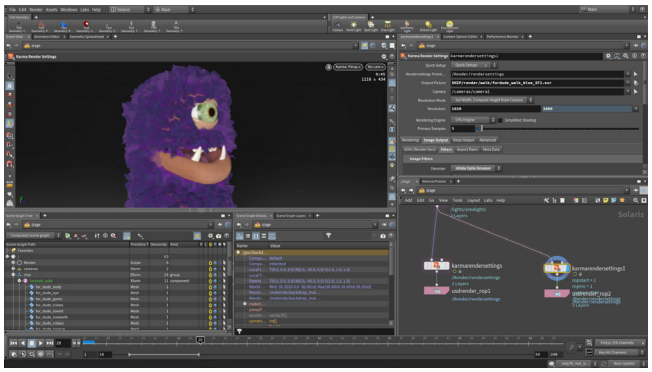
13 完了したら、**Render > Mplay > Load Disk Files** を選択し、レンダリングした画像を開いて最終的なシーケンスを確認します。

その後で別の Karma ノードを分岐させ、解像度とレンダリング設定を上げて最終的なレンダリングを行います。最初は低解像度でテストレンダリングを行い、すべてが希望通りになっていることを確認するようにしてください。



14 ヘアーやファーの設定を調整したい場合は、Scene View を LOP ネットワークにピン留めしてから、オブジェクトレベルに戻り、**Simulate Guides** ノードで **Load from Disk** チェックボックスを**オフ**にします。これで、変更はすべて最終的なレンダリングに反映されます。

ヘアーが短くなり、さらにモジャモジャになりました。好きなように調整してください。完了したら、再キャッシュ化して、**Load from Disk** を再度**オン**にします。



15 Solaris ネットワークに戻って、ファーの色を変更し、再レンダリングすることもできます。

最終レンダリングでは、**解像度**を **1920 x 1080** に上げるなど、画質の設定を変更しましょう。たとえば、**Pixel Samples** を **128**、**Light Sampling Quality** を **16** に設定するとよいでしょう。

まとめ

Houdini の **KineFX** ツールを使用して、Fur Dude キャラクタのリギング、アニメーション、レンダリングを行いました。**キャプチャリグ**を作成し、その上に**アニメーションコントロールリグ**を重ねるなど、さまざまな重要な手順を見てきました。また、キャラクタを **Houdini Digital Asset** にパック化し、一般的な**歩行サイクル**のキーフレームを設定しました。

その後、さまざまなグルーミングツールを使ってファーを追加し、**Karma** でレンダリングしました。ワークフローを一通り実行し、キャラクタのショットを作成できました。より完璧な結果を追求したい場合は、戻って手順の一部を調整し、何度もやり直してください。

冒頭で述べたように、KineFX ツールセットの現時点での主な用途は、本レッスンでは取り上げなかったリターゲティングとモーション編集です。KineFX のリギングおよびアニメーションツールは進化を続けています。このレッスンでは、今後の Houdini のプロシージャルリギングワークフローの可能性を、わずかながら、体験していただきました。

HOUDINI FOUNDATIONS

UNREAL 用 プロシージャルアセット

Houdini のノードベースのワークフローを使用してゲームアセットを作成するには、プロシージャルな考え方と作業方法を学ぶことが重要です。このレッスンでは、プロシージャルなノードとネットワークを使用してゲームアセットを作成する方法と、Houdini Engine エンジンを使用してそれらを直接 Unreal に取り込む方法を学習します。

その過程で、Houdini のユーザインターフェースのさまざまな機能を使っていきます。UI 要素がどのように連携して、ゲームアセットの構築をサポートするのかを学びましょう。

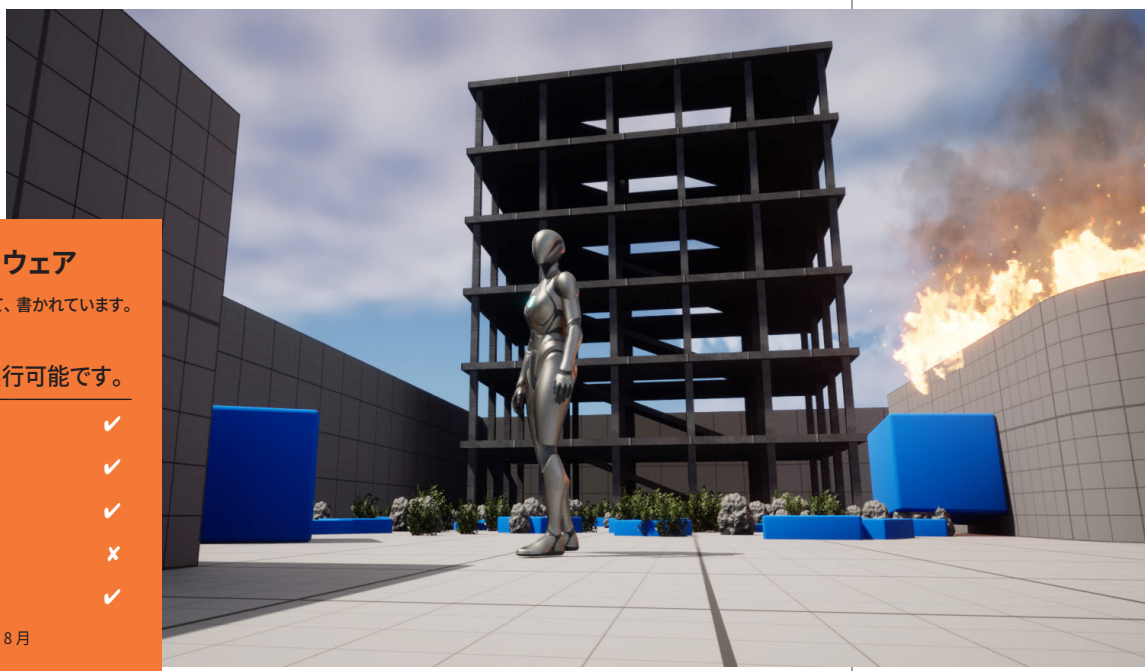
このレッスンは、**Unreal Engine 5** で実行可能です。Unreal を使ってレッスンを進めていきますが、Houdini Engine を使用して、同じアセットを Unity にインポートすることもできます。

レッスンの目標

ゲームアートとして Unreal にインポートするアセットを作成します。

学習内容

- **ノードとネットワーク**を使って、データの流れを制御する方法
- **Houdini デジタルアセット**を使用して、ソリューションをパッケージ化したり、他のユーザと共有する方法
- **Unreal Editor** に Houdini デジタルアセットをロードする方法
- オブジェクトをポイントに**インスタンス化**し、アトリビュートを使用してオブジェクトの向きやスケールを制御する方法
- **衝突ジオメトリ**を含むゲームアセットを作成し、Unreal で使用する方法
- **リジッドボディシミュレーション**を FBX ファイルとして Unreal にエクスポートする方法



使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、
以下の Houdini 製品で実行可能です。

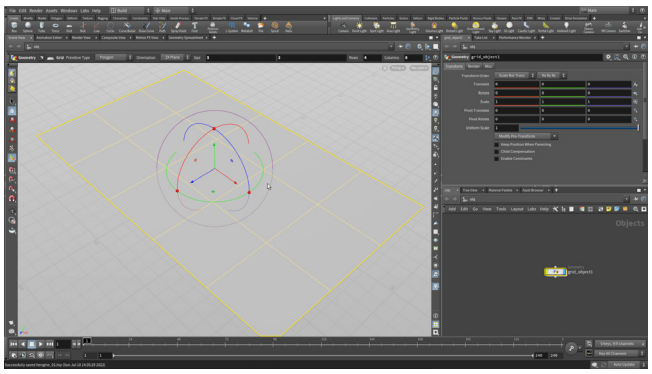
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✘
Houdini Education	✔

ドキュメントバージョン 4.0.1J | 2023 年 8 月

パート 1

シンプルなビルの作成

プロシージャルなノードネットワークを使用して、シンプルなビルを作成する方法を学習します。Scene View での操作で作成されるノード、ネットワークビューで作成されるノード、両方のノードがあります。あるノードの一部をシステム内の別のノードと接続するには、チャンネル参照を使用します。このようにして作成されたプロシージャルソリューションは、Houdini デジタルアセットにラップすることができます。



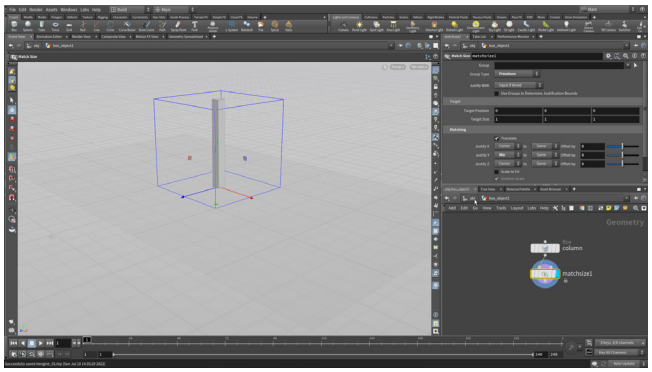
01

File > New Project を選択します。***hengine_lesson*** と名前を付け、**Accept** を押します。**File > Save As...** を選択し、ファイル名を ***hengine_01***.hip に設定したら、**Accept** をクリックして保存します。

ビューポートで、**C を押して**、**Create > Geometry > Grid** を選択します。**Enter を押して**、グリッドを原点に配置します。Scene View の上部の**オペレーションコントロール**ツールバーで、次のように設定します。

- **Size** を **5, 3** にする
- **Rows** を **4** にする
- **Columns** を **6** にする

V を押して Shading > Smooth Wire Shaded を選択します。

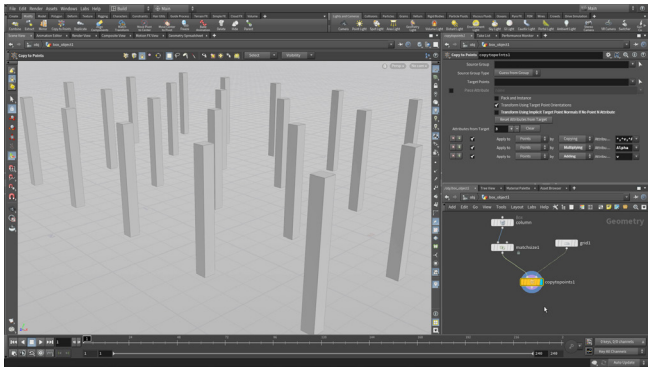


02

C を押して Create > Geometry > Box を選択します。**Enter** を押して原点に配置します。ネットワークビューで ***box_object*** を**ダブルクリック**します。これで、ジオメトリレベルでオブジェクト内に入ります。次のように設定します。

- **Size** を **0.1, 1, 0.1** にする

box ノードを ***column*** という**名前に変更**します。Scene View で Tab を押し、**Match Size** と入力していき、**Match Size** を選択します。**N を押して**ボックスを選択し、**Enter** を押します。これにより、新しいノードが追加されます。パラメータエディタの **Matching** で、**Justify Y** を **Min** に設定します。これでボックスが持ち上がり、地面の上に配置されます。

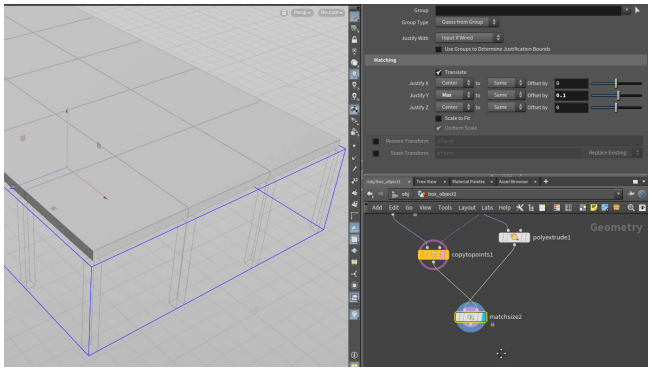


03

U を押してオブジェクトレベルに戻るか、ネットワークビューのパスバーで **obj** をクリックします。**Select** ツールをクリックし、何もない空間をクリックしてすべてのオブジェクトを選択解除します。

Modify シェルフで **Copy to Points** をクリックします。***コピーするジオメトリ***として柱を選択し、**Enter** を押します。次に、***コピー先となるポイント上のジオメトリ***としてグリッドを選択し、**Enter** を押します。

パラメータエディタで、**Transform Using Implicit Target...** を**オフ**にします。すると、柱が立ち上がります。**Pack and Instance** をオンにして、Unreal でジオメトリがインスタンス化されるようにします。**4 を押して**プリミティブ選択モードに移動します。こうすると、すべてのコーナーポイントが非表示になります。

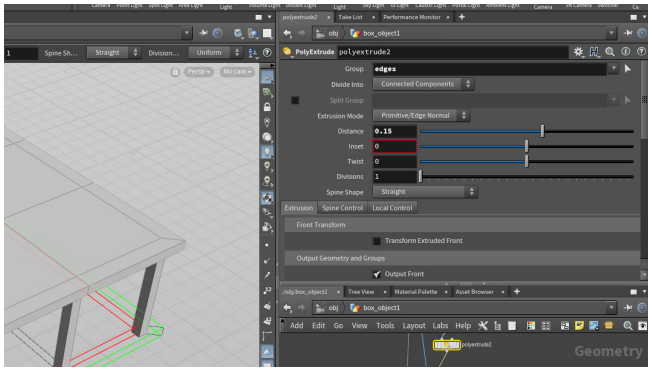


04

ネットワークビューで **Tab > PolyExtrude** を押します。それを横に配置します。grid ノードを ***polyextrude*** ノードに接続したら、**Display フラグ**を設定します。**Distance** を **0.1** に設定し、**Output Geometry and Groups** の **Output Back** をオンにします。**copytopoints** ノードに **Template フラグ**を設定します。

ネットワークに **Match Size** ノードを追加し、***polyextrude*** を 1 つ目の入力、***copytopoints*** を 2 つ目の入力に接続します。**Justify Y** を **Max to Same**、**Offset** を **0.1** に設定します。これで、押し出された形状が柱の上に配置されます。

Merge ノードを追加し、それに ***copytopoints*** と ***matchsize*** を接続します。

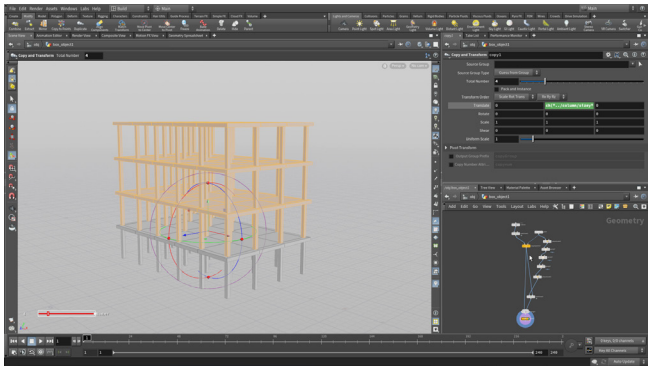


05

ネットワークビューで、**Tab > Group** を押します。それを、***polyextrude*** と ***matchsize*** の間に接続します。**Group Name** を ***edges*** に変更します。

Alt ドラッグして ***group*** ノードをもう 1 つ作り、**Display フラグ**を設定します。**Group Name** は ***edges*** のままで、**Initial Merge** を **Subtract from Existing** に設定します。**Base Group** で **Enable** をオフにし、**Keep by Normals** で **Enable** をオンにします。**Direction** を **0, 1, 0**、**Spread Angle** を **0** に設定します。このノードを **Alt ドラッグ**してコピーを作成し、チェーンに接続します。**Direction** を **0, -1, 0** に変更します。これで、***edges*** グループ内にあるのはボックスの側面のみとなりました。

PolyExtrude ノードを追加します。**Group** を ***edges*** に、**Distance** を **0.15** に設定します。

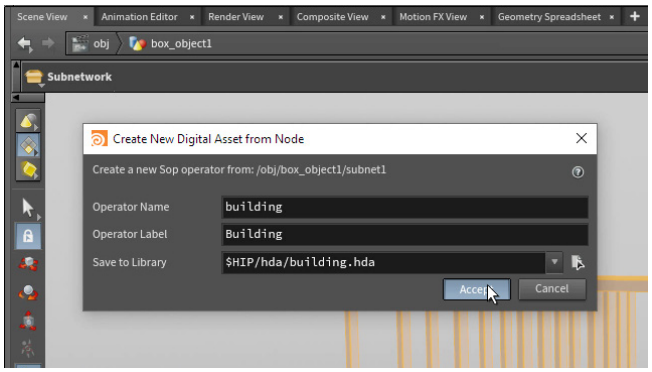


06

merge ノードに **Display フラグ**を設定し、ビルの 1 階をすべて表示します。***copytopoints*** ノードの **Template** フラグをオフにします。

Scene View で、**N を押して**すべてを選択してから、**Tab > Copy and Transform** を選択します。***column*** ノードを選択し、**Center Y** を **RMB クリック**して **Copy Parameter** を選択します。***copy*** ノードに移動し、**Translate Y** を **RMB クリック**して、**Paste Relative References** を選択します。エクスプレッションに **+ 0.1** を追加します。

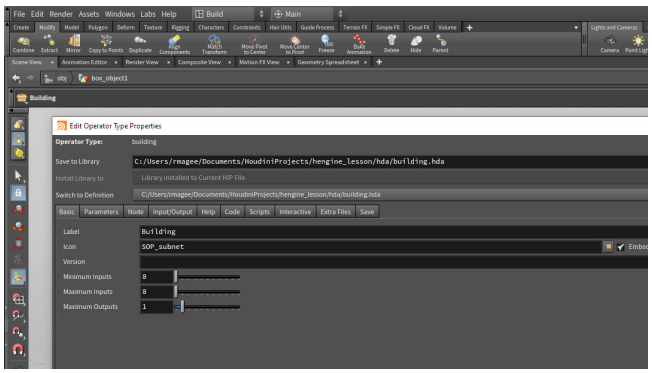
ここで、**Total Number** を 4 に増やして、3 フロア追加します。



07

ネットワークエディタですべてのノードを選択します。**Assets** メニューから、**New Digital Asset From Selection** を選択します。これにより、ネットワークがサブネットワークに折り畳まれ、そのサブネットノードを使用してデジタルアセットが作成されます。

Operator Name を ***building*** にすると、**Operator Label** が ***Building*** に変更されます。**Save to Library** の右端のボタンをクリックします。***Locations*** サイドバーで、$HIP/ をクリックしてから ***hda*** ディレクトリをダブルクリックします。**Accept** を押したら、再度 **Accept** を押してアセットをディスクに保存します。



08

Edit Type Properties ウィンドウが開きます。このパネルで、アセットのユーザインターフェースを構築します。このウィンドウはまた後で使用します。**Accept** をクリックして、このウィンドウを閉じます。

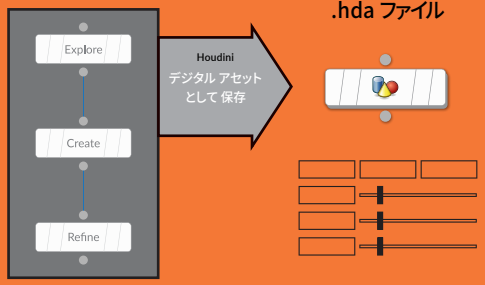
ネットワークビューのノードを building という**名前に変更**します。

アセットを構築するのに使用したノードは、保存した後もアセットの一部であり続けます。このため、Unreal ゲームレベルでアセットを使い始めた後も、引き続き変更を加えることができます。

HDA ファイルとは？

Houdini ノードとネットワークは、**Houdini デジタルアセット**と呼ばれる単一のノードにカプセル化でき、それを利用してテクニックを同僚と共有できます。アセットは、ディスク上の **.hda** ファイルと呼ばれるファイルに保存されます。

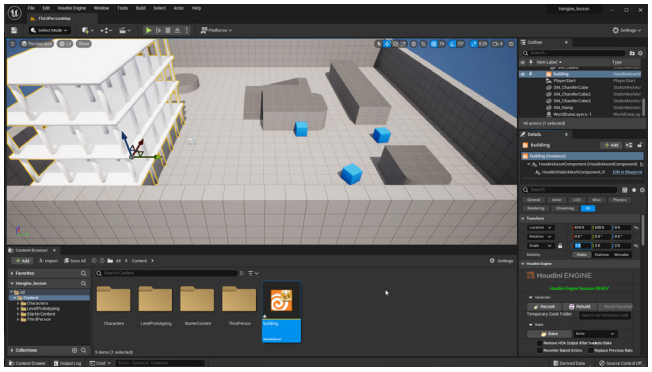
Houdini の旧バージョンで作成したアセットファイルは、**.otl** (Operator Type Library) という別の拡張子が付いていますが、どちらのタイプのファイルも同じように使用できます。



パート 2

Unreal にアセットをインポート

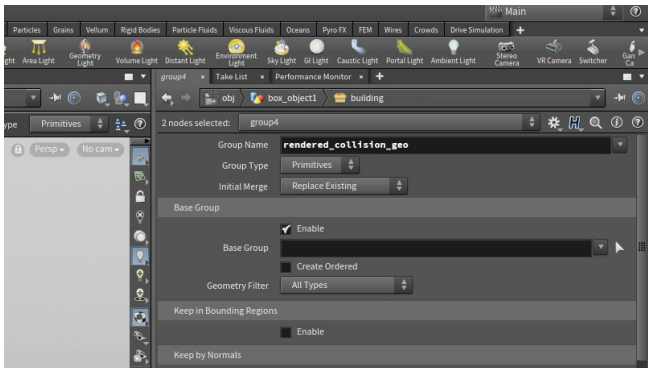
デジタルアセットファイルがディスクに保存され、Unreal Engine にインポートできる状態になりました。これが可能なのは、Houdini Engine プラグインが 2 つのアプリケーションを繋いでいるからです。Unreal にロードされる Houdini デジタルアセットは、Houdini を使用して内部的にクックされます。**このレッスンを完了するには、Houdini Engine for Unreal プラグインをインストールする必要があります（Sidefx.com/unreal をご覧ください）。*Houdini Apprentice で Houdini Engine を実行することはできません。***



01

UNREAL で – ブループリントをサポートする**サードパーソンテンプレート**をセットアップします。***TextRenderActor*** を削除します。**Content Browser** を開き、**Import** ボタンをクリックします。**Content Browser** をドッキングさせておくとよいでしょう。Houdini プロジェクトに移動し、***building*** アセットを指定します。**Open** をクリックします。

Scale を **2, 2, 2** に設定し、アセットをコーナーに移動します。**Play** ボタンを押すと、ゲームプレイ中にビルを確認できます。

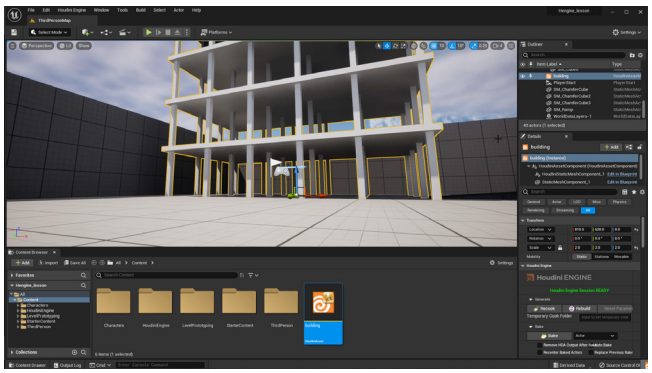


02

HOUDINI で – ジオメトリに法線と衝突を追加するには、ネットワークにノードをいくつか追加する必要があります。

building ネットワークに入り、***polyextrude*** と ***matchsize*** ノードの間に **Normal** ノードを追加します。ジオメトリに法線が追加され、Unreal 内で適切に表示されるようになります。***normal*** ノードの下に **Group** ノードを追加し、**Group Name** を ***rendered_collision_geo*** に設定します。これにより、ジオメトリが衝突ジオメトリになります。

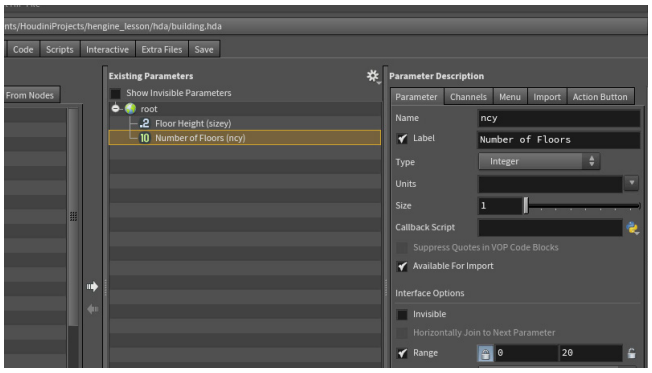
これら 2 つのノードを**コピーアンドペースト**して、新しいノードを ***column*** ノードの下に挿入します。



03

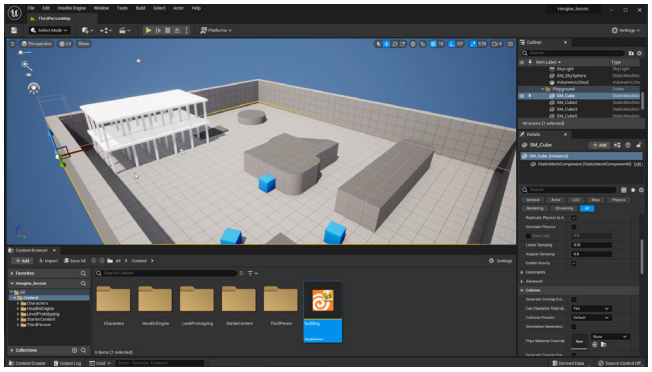
UNREAL で – ***building*** アセットの **Details** パネルの Generate セクションで、**Rebuild** を押します。**Play** を押して、シーンを確認します。

法線が適切に動作し、通り抜けようとすると、柱と衝突するようになりました。



04

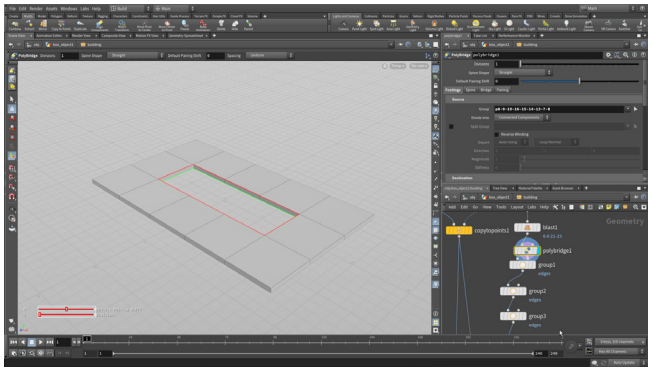
HOUDINI で – **Assets > Edit Asset Properties > Building** を選択します。Type Properties ウィンドウで Parameters タブをクリックします。***column*** ノードを選択し、**Size Y** パラメータを Parameter タブにドラッグします。それを **Floor Height** という**名前に変更**します。次に、***copy*** ノードから **Total Number** パラメータをドラッグし、名前を **Number of Floors** に変更します。**Accept** をクリックします。



05 **UNREAL で** – ビルのアセットの **Details** パネルで **Rebuild** を押します。下にスクロールすると、**Floor Height** と **Number of Floors** のパラメータを確認できます。

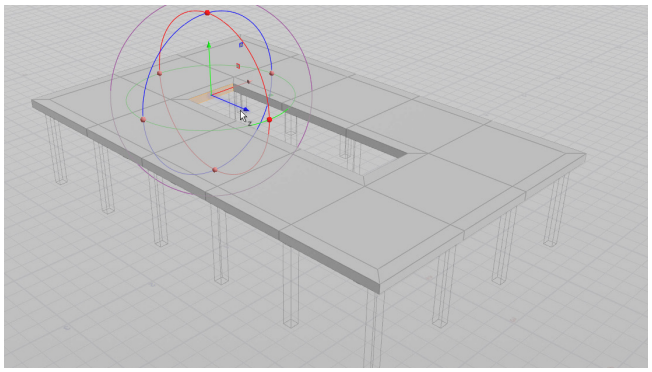
Floor Height を **1.2**、**Number of Floors** を **2** に設定します。

Play を押してアセットを見て回ったり、変更をレビューします。



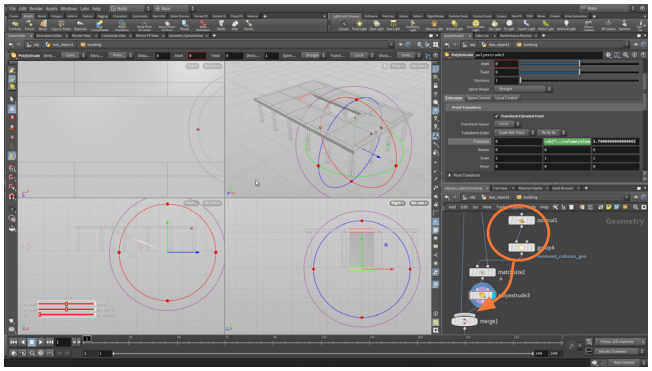
06 **HOUDINI で** – HOUDINI に戻り、床板である ***polyextrude*** ノードに **Display フラグ**を設定します。**4 を押して**プリミティブ選択にします。上部の中央にある 3 つのプリミティブを Shift を押しながら選択したら、そのままタンブルして、スラブの下部の 3 つのプリミティブを選択します。**Delete** を押します。これにより、**Blast** ノードが追加されます。

3 を押してエッジ選択にします。穴の下部のエッジを**ダブルクリック**し、ループ全体を選択します。**Tab > PolyBridge** に移動して **Enter** を押します。穴の上部のエッジを**ダブルクリック**し、**Enter** を押します。これにより、穴の内側にジオメトリが追加されます。



07 ***matchsize*** ノードに **Display フラグ**を設定します。突出部のある新しい穴が表示されます。

4 を押してプリミティブ選択にします。**Select** ツールにして、穴の側面を選択します。選択されているようには見えませんが、実際にはされています。**Tab > PolyExtrude** を押します。Extrusion で、**Transform Extruded Front** をオンにします。フェースを引き出し、下げます。



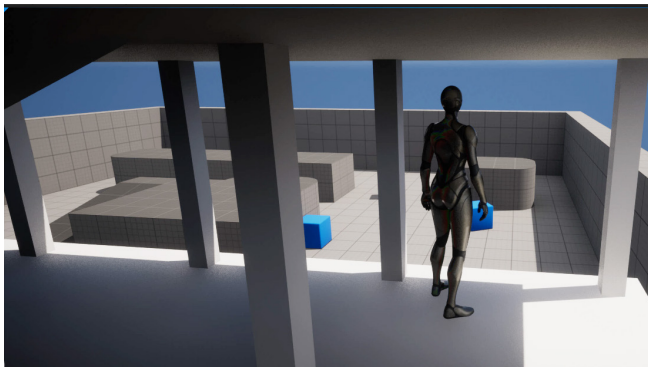
08 ***column*** ノードの **Size Y** チャンネルをコピーし、**Paste Relative Reference** で新しい ***polyextrude*** ノードの **Translate Y** にペーストします。エクスプレッションに –(マイナス)記号を追加し、0.1 を減算します。エクスプレッションは次のようになります。

```
-ch("../column/sizey")-0.1
```

Translate Z を **2.7** に設定します。

normal と ***group*** ノードを、***polyextrude*** ノードの後に移動します。こうすることで、新しい押し出しが衝突ジオメトリになり、斜面が適切に動作します。

output ノードに **Display フラグ**を設定します。**Assets > Save Asset > Building** を選択し、変更をディスク上の HDA ファイルに保存します。

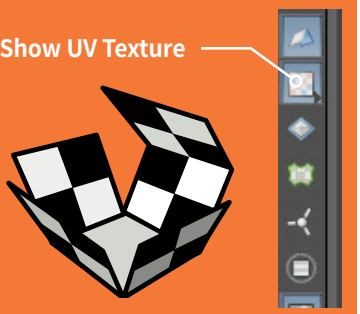


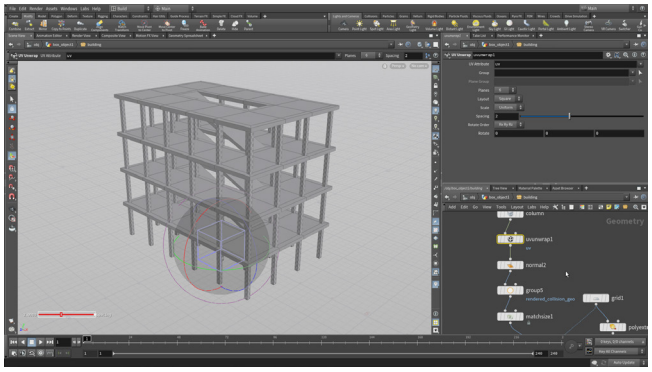
09 **UNREAL で** – **Rebuild** を押し、**Floor Height** を **1**、**Number of Floors** を **6** に設定します。**Play** を押し、斜面を上がってビルを見て回ります。

UV

Houdini には、ジオメトリレベルで UV のセットアップと管理を行えるさまざまなノードがあります。このレッスンでは、**UV Unwrap** と **UV Transform** を使用してビルに UV を追加します。これらのノードは、ビルのモデリングに使用したようなシンプルな形状でうまく機能します。より複雑な形状には、**UV Flatten** や **UV Layout** などのツールを使用します。

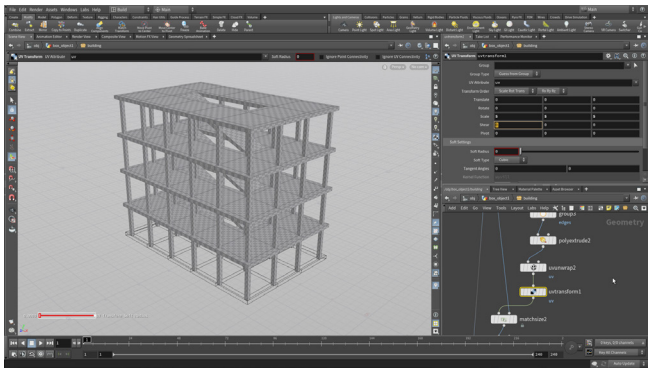
UV を配置すると、ジオメトリに UV グリッドが表示されます。グリッドを非表示にしたい場合は、**Display Options** バーの **Show UV Texture** ボタンをクリックすると、オンとオフを切り替えられます。





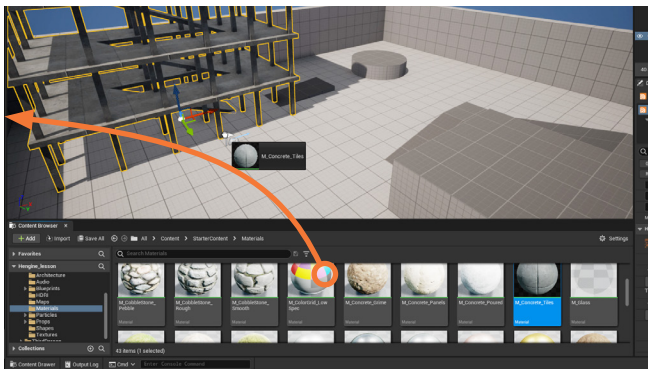
10 **HOUDINI で** – ゲームエディタで使用するには、ジオメトリに UV を追加する必要があります。ネットワークの ***column*** ノードがある部分に移動します。**Tab > UV Unwrap** を押し、***column*** ノードと ***normal*** ノードの間にノードを配置します。

ネットワークの ***2 つ目の polyextrude*** ノードがある部分に移動します。**Tab > UV Unwrap** を押し、***polyextrude*** ノードと ***normal*** ノードの間にノードを配置します。



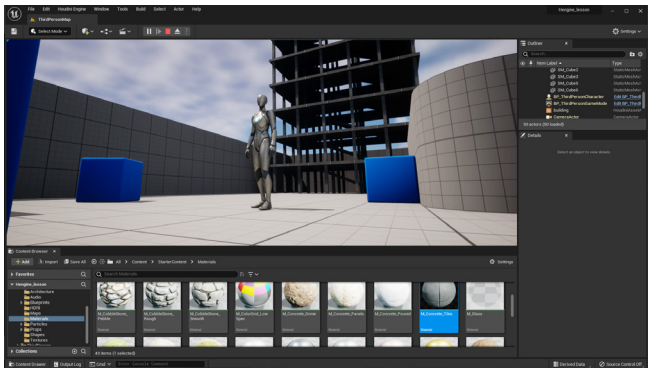
11 柱に UV グリッドが表示されます。柱と床のグリッドのスケールが一致していません。**Tab > UV Transform** を押し、***uvunwrap*** ノードと ***matchsize*** ノードの間にノードを配置します。**Scale** を **5, 5, 5** に設定します。これで、同じような UV になります。

Assets > Save Asset > Building を選択し、変更をディスク上の HDA ファイルに保存します。



12 **UNREAL で** – **Rebuild** を押します。マテリアルが追加されるまで、UV は表示されません。

Content Browser で、**Content > Starter Content > Materials** に移動します。***Building*** アセットが選択されていることを確認し、***M_Concrete_Tiles*** などのマテリアルを Scene View の**ビルのジオメトリ**に追加します。柱と床板の両方にドラッグしてください。

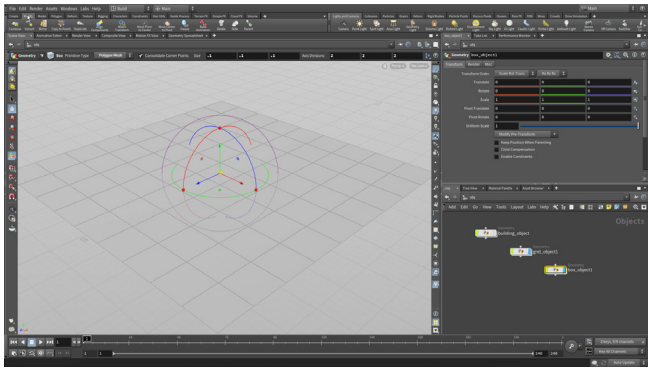


13 **Play** を押してレベルを歩き回り、テクスチャ付きのビルを探検しましょう。

このアセットは、1 つのレベルで何度も使用して、階数や高さの違うビルを作れます。また、さらに多くのパラメータをアセットにプロモートすれば、より詳細にコントロールすることも可能です。デジタルアセットのネットワークやパラメータインターフェースに変更を加えると、レベル上のすべてのアセットが即時に更新されます。

パート 3
ポイントへのコピー

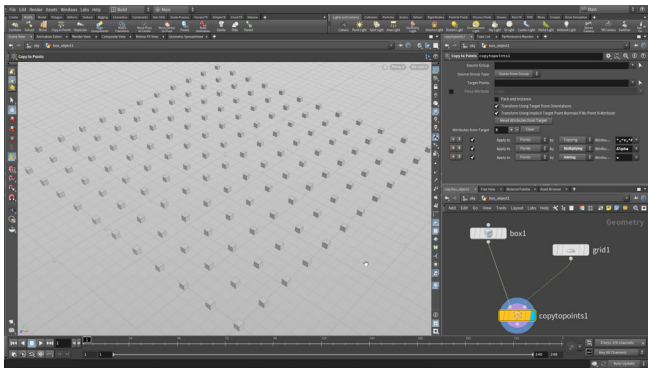
このパートでは、いくつかのボックスを別のグリッドのポイントにコピーします。次に、ポイントをランダム化してより有機的なルックにしたり、ランダムなアトリビュートを追加してボックスを回転およびスケールすることで、形状の分布にバリエーションを持たせます。このようにしてもう 1 つプロシージャルシステムを作成し、後ほど Houdini デジタルアセットにラップします。



01

HOUDINI で – オブジェクトレベルに移動して、ビルを非表示にします。ビューポートで、**C** キーを押して Radial メニューを表示し、**Create > Geometry > Grid** を選択します。**Enter** を押して原点に配置します。このグリッドサーフェス上のポイントを使用して、ジオメトリをインスタンス化していきます。

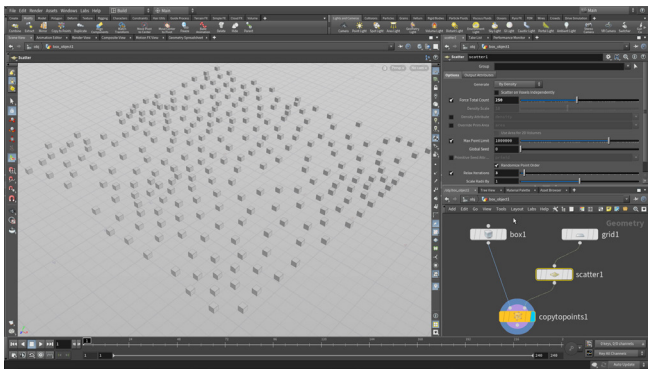
同じ Radial メニューを使用して **Create > Geometry > Box** を選択し、再度 **Enter** を押して原点に配置します。ビューポートの上部の**オペレーションコントロール**ツールバーで、**Size** を **0.1 0.1 0.1** に設定します。このジオメトリを、グリッド上のポイントにコピーしていきます。



02

ボックスを選択したまま、**Modify** シェルフに移動して **Copy to Points** を選択します。***グリッド***を選択して、**Enter** を押します。これで、ボックスがすべてのグリッドポイントにコピーされました。

パラメータを編集して、システムのルックを調整します。***grid*** ノードを選択し、**Size** を **6, 6**、**Rows** と **Columns** を **12** に設定します。グリッドが増え、ポイントの数も増加しますが、ボックスはすべてのポイントにコピーされていません。***copytopoints*** ノードをクリックすると、**Target Points** が元のグリッドのポイントに合わせて **0-99** に設定されているのが分かります。**0-99** を削除して、グリッド全体にボックスをコピーするようにします。



03

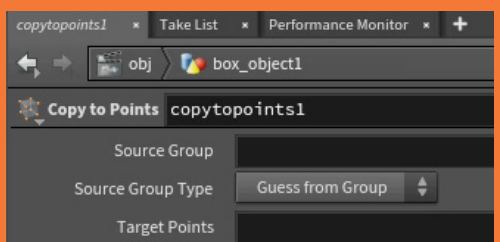
ネットワークエディタで、**Tab > Scatter** を選択します。ネットワークエディタの ***grid*** と ***copytopoints*** ノードの間に ***scatter*** ノードを配置します。すると、このノードが自動的にネットワークに接続され、ボックスが新しいポイントにコピーされます。

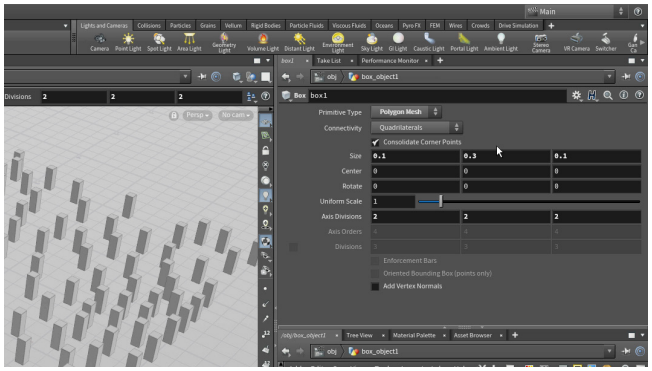
Force Total Count を **250** に設定します。**Relax Iterations** を変更し、ポイントの配置を調整します。scatter ノードを使用すると、元のグリッドポイントよりも有機的な配置にすることができます。

SOURCE GROUP

Scene View でモデリングする場合、選択したジオメトリは、ツールに応じて番号付きのポイントまたはプリミティブとして、**Source Group** フィールドに設定されます。フィールドが空の場合、ツールは**すべて**のポイントまたはプリミティブに作用します。

ジオメトリレベルでツールを使用する場合、ツールを使用する前に **Select All (N キー)** を選択すると、このフィールドは空になります。ここでは 0-99 のポイントが設定されていますが、これはオブジェクトレベルで ***copytopoints*** をセットアップしたからです。

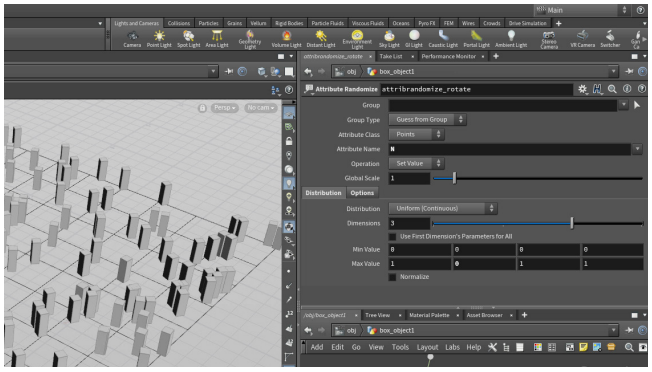




04

ネットワークビューで Tab を押して、**Match Size** と入力していき、**Match Size** を選択します。このノードを ***box*** ノードと ***copytopoints*** ノードの間に配置します。パラメータエディタの **Matching** で、**Justify Y** を **Min** に設定します。

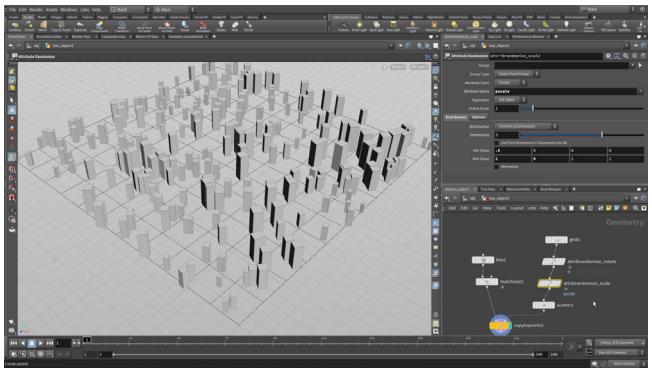
box ノードで、**Size Y** を **0.3** に設定してエクスプレッションをテストします。すべてのボックスが地面の上にあることが確認できます。



05

ネットワークエディタで、Tab を押して **rand...** と入力していき、**Attribute Randomize** ツールを選択します。***attributerandomize*** ノードを ***grid*** と ***scatter*** ノードの間に配置します。デフォルトのアトリビュートがカラー (Cd) のため、最初はボックスがランダムなカラーで表示されます。

Attribute Name を **N** に設定します。これによりアトリビュートが法線方向に変わり、すべてのボックスが異なる方向を向くようになります。**Max Value Y** を **0** に設定し、ランダム化を X 方向と Z 方向に制限します。ノードの名前を ***attribrandomize_rotate*** に変更します。



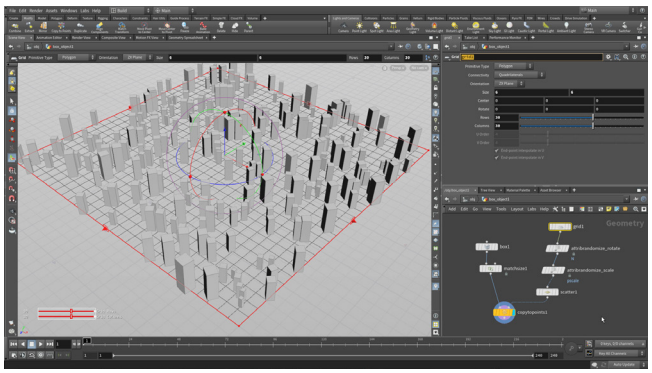
06

Alt キーを押しながら ***attribrandomize_rotate*** ノードをドラッグし、コピーを作成します。それを、***attribrandomize_rotate*** と ***scatter*** ノードの間にドロップします。ノードの名前を ***attribrandomize_scale*** に変更します。

次のように設定します。

- **Attribute Name** を ***pscale*** にする
- **Min Value** を **0.5** にする
- **Max Value** を **2** にする

これにより、グリッド上のボックスのサイズがいい具合にばらつきます。



07

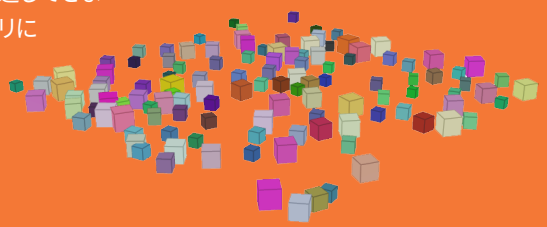
grid ノードを選択し、**Rows** と **Columns** を **30** に上げます。グリッドのポイント数が増え、さらにランダムにポイントがばら撒かれるようになります。

作業内容を**保存**します。

アトリビュートの仕組み

ジオメトリに割り当てられたアトリビュートは、ネットワークチェーンを通じてさまざまなノードに渡されます。このネットワークの場合、グリッドジオメトリに割り当てられたアトリビュートは、ばら撒かれたポイントに渡されて、コピーされたジオメトリに作用しています。Houdini では、このようにしてデータの流れがコントロールされています。

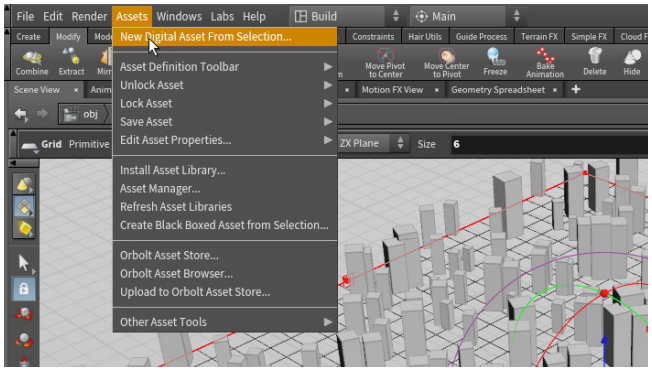
アトリビュートは最初グリッド上のポイントに割り当てられるため、ポイント数が多いほど、アトリビュート値がランダムになります。



パート 4

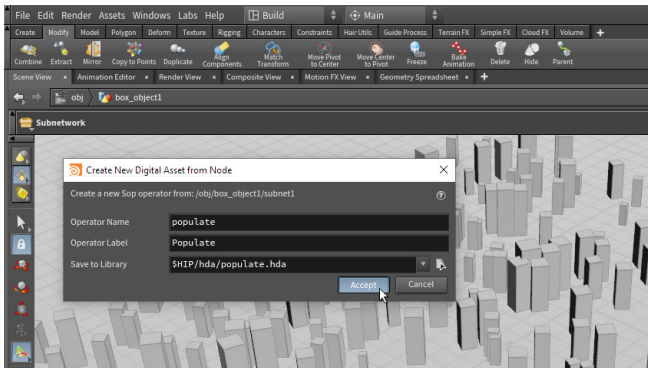
Houdini デジタルアセットをもう 1 つ作成

このパートでは、デジタルアセットを 1 つ作成し、Unreal でシステムをテストします。ビルと同じように、ネットワークをラップして、結果を HDA ファイルとしてディスクに保存します。一部のパラメータをプロモートして、グリッドサイズ、ポイント数、リラックス化をコントロールできるようにします。そのパラメータは、Unreal でも使用できます。



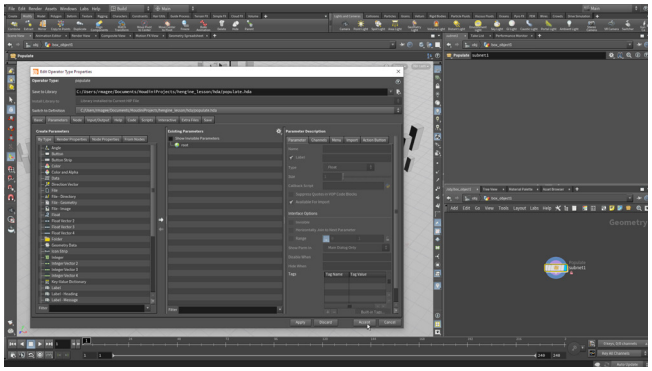
01 ネットワークエディタですべてのノードを選択します。**Assets** メニューから、**New Digital Asset From Selection** を選択します。これにより、ネットワークがサブネットワークに折り畳まれ、そのサブネットノードを使用してデジタルアセットが作成されます。

アセットを構築するのに使用したノードは、保存した後もアセットの一部であり続けます。このため、ゲームレベルでアセットを使い始めた後も、引き続き変更を加えることが可能です。



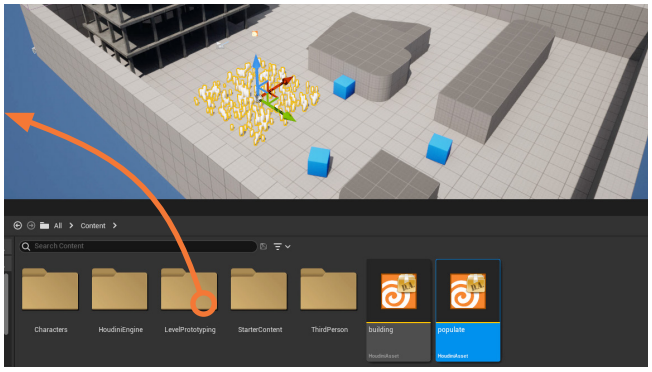
02 **Operator Name** を ***populate*** にすると、**Operator Label** が ***Populate*** に変更されます。**Save to Library** の右端のボタンをクリックします。***Locations*** サイドバーで、$HIP/ をクリックしてから hda ディレクトリをダブルクリックします。**Accept** を押したら、再度 **Accept** を押してアセットをディスクに保存します。

これにより、このシーンで参照される新しい Houdini デジタルアセットファイル (.hda) が作成されます。別の Houdini シーンで参照することも、Houdini Engine を使用して Unreal など別のアプリケーションで参照することもできます。



03 **Edit Type Properties** ウィンドウが開きます。このパネルで、アセットのユーザインターフェースを構築します。このウィンドウはまた後で使用します。**Accept** をクリックして、このウィンドウを閉じます。

Houdini デジタルアセットのプロシージャルな性質を維持するために、ハイレベルなインターフェースを構築して、ネットワーク内のノードにアクセスできるようにします。レッスンの後半では、このアセットのインターフェースを増やします。



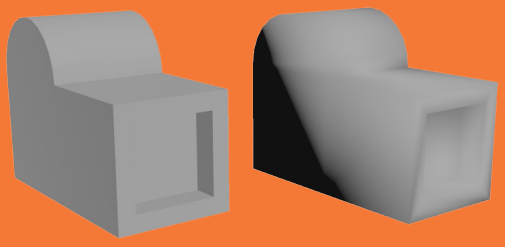
04 **UNREAL で** – Content Browser で、***Content*** ディレクトリに戻ります。**Import** をクリックし、現在のプロジェクトディレクトリの ***populate.hda*** ファイルを指定します。そのアセットを Content Browser から 3D ワークスペースにドラッグします。

Play を押してアセットを見て回ります。アセットはありますが、特に変わったところはありません。**Esc** を押してアセットの UI に戻ります。

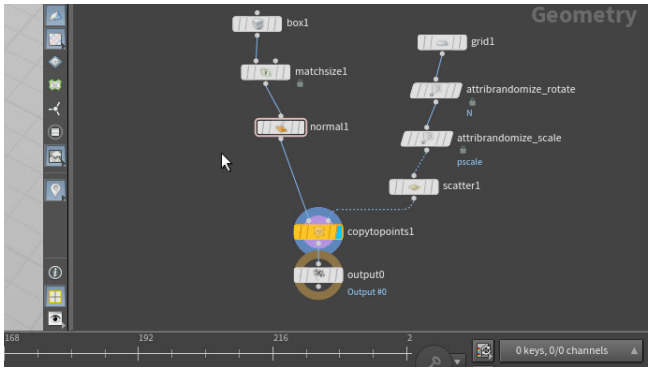
頂点法線

デフォルトでは、ボックスなどの Houdini オブジェクトには、ポイント法線はありますが頂点法線がありません。適切な頂点法線を設定するには、**Normal** ノードを追加し、cusp 値を使用してハードに見せたいエッジと、ソフトに見せたいエッジを決定する必要があります。

Unreal などのゲームエディタは、適切な表示に頂点法線を必要としますが、これは既存のネットワークで簡単にセットアップできます。

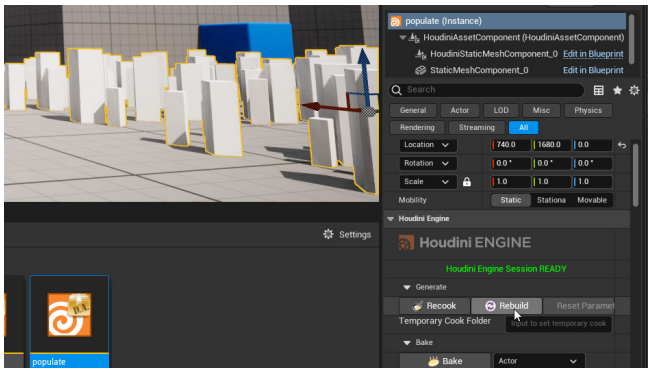


法線あり　　法線なし



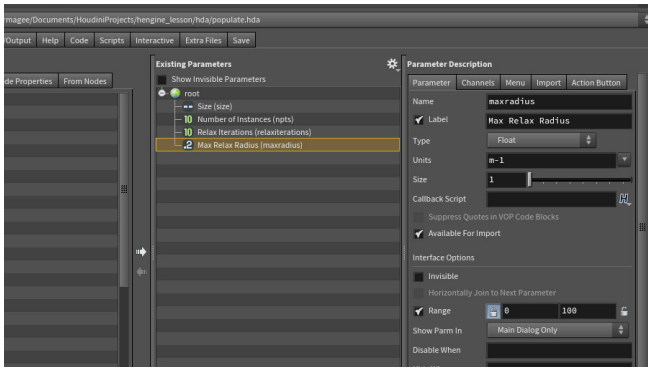
05 **HOUDINI で** – Unreal でアセットについて気になるのは、ボックスの角がくっきりしていないことです。これを修正するには、Houdini に戻り、ネットワークエディタで **Tab** を押して **norm...** と入力していき、**Normal** ツールを選択します。

matchsize ノードの直後に **Normal** ノードを追加します。**Assets** メニューから、**Save Asset > Populate** を選択します。これにより .hda ファイルに変更が保存されるため、.hda ファイルをロードしたすべての人が更新されたアセットを使用できるようになります。ここでは、Unreal 内でアセット定義を更新し、正しい法線を表示させます。



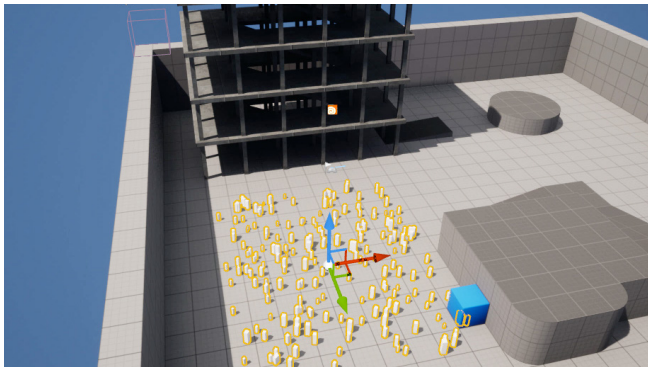
06 **UNREAL で** – Details パネルの Houdini Asset で、**Cooking Actions** セクションを開きます。**Rebuild** ボタンをクリックし、変更を確定します。これにより、Houdini 内で加えられた変更が、Unreal シーンで適切に更新されます。

アセットの法線は適切になりましたが、プロシージャルネットワークをコントロールすることができません。Houdini で使用できるインターフェースを作成するため、一部のパラメータをアセットの内部からトップレベルにプロモートします。



07 **HOUDINI で** – **Assets > Edit Asset Properties > Populate** を選択します。**Parameters** タブをクリックします。ネットワークエディタで、***grid*** ノードをクリックします。パラメータエディタから、**Size** パラメータを **Existing parameters** リストの root にドラッグします。

ネットワークエディタで、***scatter*** ノードをクリックします。パラメータエディタから、**Force Total Count** パラメータを root にドラッグします。Parameter Description で、**Label** を **Number of Instances** に変更します。**Relax Iterations** と **Max Relax Radius** をドラッグして、これらのパラメータをアセットに追加します。**Accept** をクリックすると、これらの新しいパラメータがアセットに保存されます。



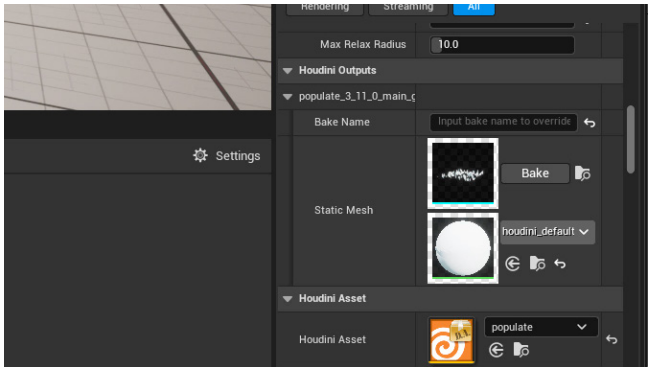
08 **UNREAL で** – **Rebuild** ボタンをクリックして変更を確定します。パラメータエディタにプロモートしたパラメータが表示されています。**Size** と **Number of Instances** を変更して、ボックスのグリッドのルックにどう影響するかを確認します。

これで、アセットのプロシージャルな性質を体験し、それぞれのレベルに特有のアセットを作成できるようになりました。このレベルで ***populate*** アセットを複数追加すると、.hda ファイルの同じアセットを参照しながら、それぞれのアセットに独自の設定を持たせることができます。このアセットを複数のレベルで使用して、複数のアーティストが作業することも可能です。

パート 5

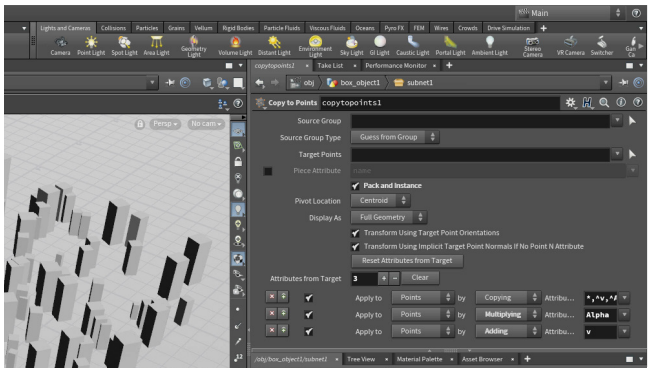
インスタンス化のセットアップ

Houdini でインスタンス化を適切にセットアップすると、ポイントにコピーしたデフォルトの形状を、別の Unreal プロップに置き換えられます。複数のプロップを追加して、デフォルト形状の代わりにランダムに分布させることができます。ここでは、インスタンス化が実際に適切にセットアップされていることを確認し、それを利用して別のプロップをシステムに追加していきます。



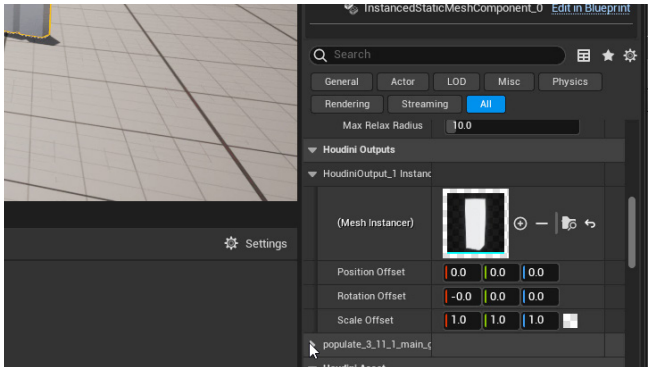
01 **UNREAL で** – Details パネルで **Houdini Outputs** を開きます。すべてのボックスが単一のメッシュとしてインポートされていることが分かります。つまり、まだインスタンス化は使用されていません。Houdini は、ボックスをポイントにコピーし、結果のジオメトリを Unreal 用に出力しています。

これは、ゲームプレイにとってあまり効率的ではありません。セットアップを少し変えてインスタンス化し、このツールが適切に機能するようにする必要があります。



02 **HOUDINI で** – ***copytopoints*** ノードを選択し、**Pack and Instance** をオンにします。**Assets** メニューから、**Save Asset > Populate** を選択します。

パックプリミティブを使用することで、***copytopoints*** ノードに接続されているボックスジオメトリが単一のプリミティブとして扱われます。これで Houdini でインスタンス化がセットアップされ、Houdini Engine プラグインを使用してこのアセットがエディタにロードされると、Unreal でのインスタンス化がトリガされるようになりました。



03 **UNREAL で** – **Rebuild** ボタンをクリックして変更を確定します。**Houdini Outputs** セクションに移動します。1 つのボックスのみがインポートされ、ポイントにインスタンス化されていることが分かります。そして、前にセットアップしたアトリビュートに基づいて、回転およびスケールされます。

デフォルトのボックスを、Unreal 環境で他のジオメトリに置き換えることができました。エディタでの柔軟性が高く、さまざまなオブジェクトをシステムに追加できるので、複数のポイントに配置したい場合にはこの方法が最適です。

HOUDINI のパックプリミティブ

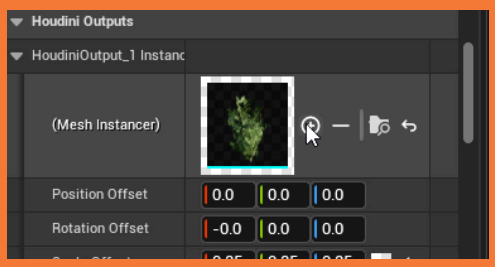
Houdini では、パックプリミティブを使用すると、ビューポート表示やレンダリング向けに効率的にインスタンスを管理できます。コピーノードに接続されているジオメトリが単一のプリミティブにパックされ、インスタンスのように扱われます。このレッスンで示されているボックスのコピーであれば、パック化を実行すると、プリミティブが 1,500 から 250 に減少します。Houdini Engine for Unreal プラグインを使用すると、パックプリミティブは Unreal インスタンスとして認識されるため、より効率的なゲームプレイが可能になります。

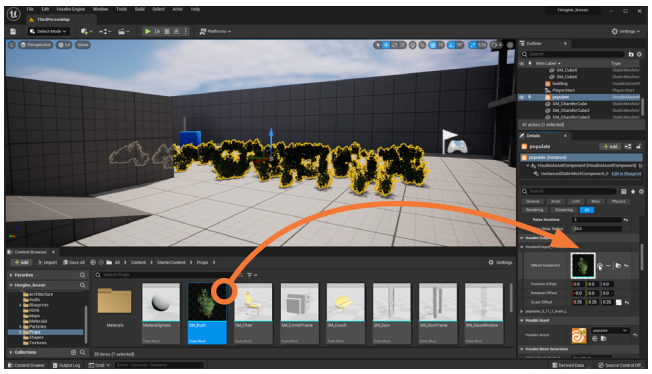
Points	2,000	Center	-0.0227511, 0.286
Primitives	1,500	Min	-3.12244,
Vertices	6,000	Max	3.07694, 0.573
Polygons	1,500	Size	6.19938, 0.573

Points	250	Center	-0.0227511,
Primitives	250	Min	-3.12244, -1.0
Vertices	250	Max	3.07694,
Packed Geos	250	Size	6.19938,

UNREAL のインスタンス

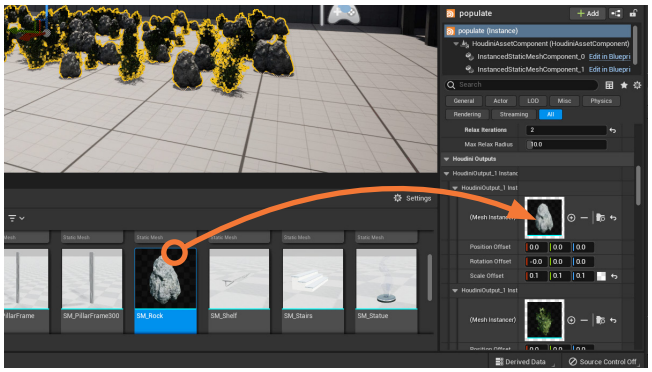
Houdini Engine プラグインがパックプリミティブを検出すると、Details タブに Houdini Output Instancer が作成されます。そこには、入力ジオメトリと、インスタンスを回転およびスケールするためのパラメータが含まれています。このインスタンスを、Unreal のコンテンツウィンドウのジオメトリと置き換え、適切なサイズに変えることができます。+（プラス）記号を使用して、インスタンス化する入力を追加すれば、同じシステム内でさらにバリエーションを作成できます。





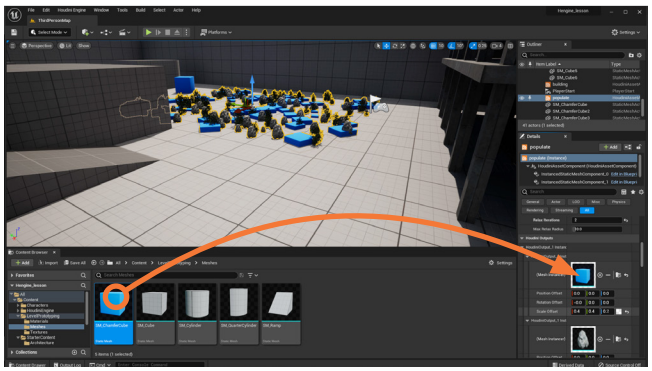
04 **Houdini Outputs** セクションに移動し、***HoudiniOutput1 Instancer*** を展開します。このボックスのインスタンスを、Unreal 内のコンテンツで置き換えます。

Content Browser で、**StarterContent > Props** を開きます。***SM_Bush*** プロップを **Houdini Outputs** 上にドラッグします。3 軸すべての **Scale Offset** を **0.25** に設定します。ジオメトリが populate アセットのポイントにインスタンス化され、ボックスと同じように回転およびスケールされます。



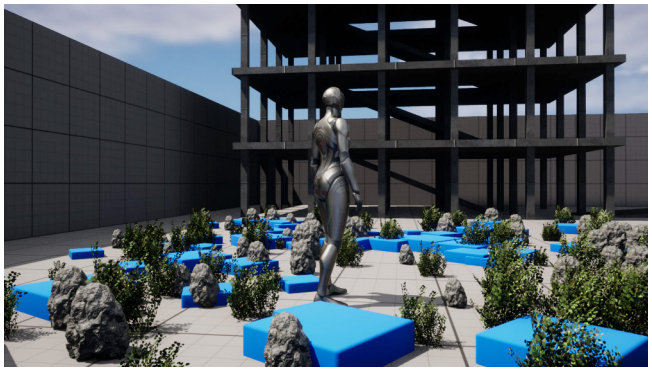
05 インスタンスオブジェクトの横にある **+（プラス）記号**をクリックします。これで 2 つ目が追加されます。***SM_Rock*** プロップを新しい **Houdini Instanced Input** 上にドラッグします。3 軸すべての **Scale Offset** を **0.1** に設定します。

Details パネルで、**Houdini Engine** セクションにスクロールして、**Bake** ボタンをクリックします。Outliner で、***HoudiniAssetActor*** にスクロールします。目のアイコンをクリックして非表示にして、デジタルアセットに集中できるようにします。**Play** を押してシーンを歩き回り、実際に置き換えられたインスタンスを確認します。



06 茂みのオブジェクトの横にある **+（プラス）記号**をクリックします。これで 3 つ目が追加されます。**Content > LevelPrototyping > Meshes** フォルダに移動します。***SM_ChamferCube*** を新しい **Houdini Instanced Input** 上にドラッグします。**Scale Offset** を **0.4, 0.4, 0.2** に設定します。

インスタンス化されたオブジェクトをさらに追加し、バリエーションを増やしたり、サイズを変えたりします。インスタンス同士を少し離したい場合は、**Relax Iterations** パラメータを使用します。

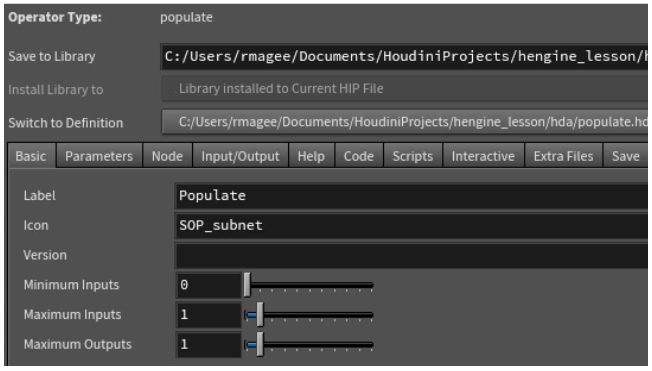


07 ***building*** アセットの **Details** パネルの Generate セクションで、**Rebuild** を押します。Bake セクションで、**Replace Preview Bake** をオンにし、**Bake** ボタンをクリックします。

Play を押してシーンを歩き回り、インスタンス化されたジオメトリを確認します。衝突ジオメトリが適切にセットアップされている、キューブと衝突するようになりました。インスタンス化したオブジェクトには、衝突ジオメトリを適切にセットアップしましょう。

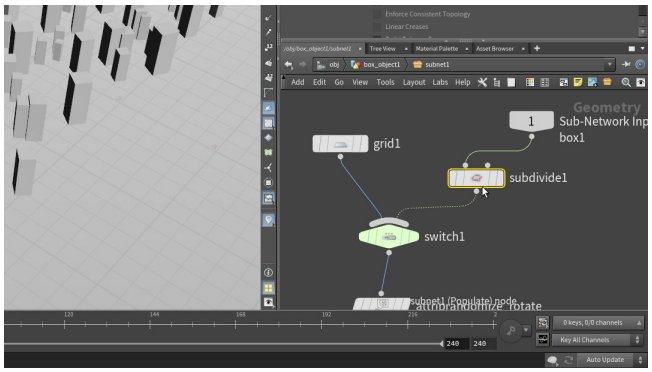
パート 6
ジオメトリを使用してアセットを駆動

ここまでは、このアセットが populate アセットに入力ジオメトリを供給してきました。Unreal シーンの既存のジオメトリを使用して、ジオメトリをインスタンス化することも可能です。ここでは、Unreal ジオメトリを受け取るアセットに、入力ノードを追加します。レベル上のオブジェクトを使用できるため、プロシージャルアセットと既存のゲームアートをよりよいかたちで統合できます。



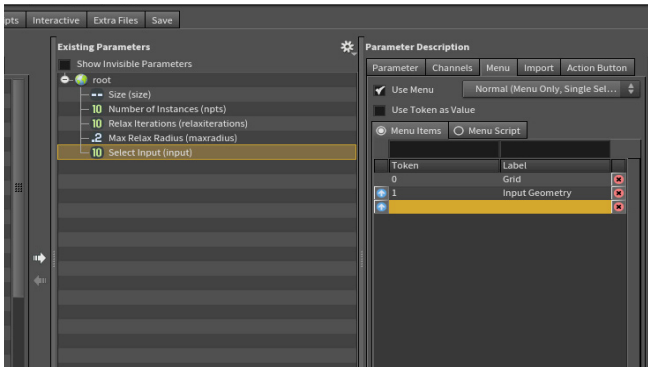
01 **HOUDINI で** – **Assets > Edit Asset Properties > Populate** を選択します。**Basic** タブで、**Maximum Inputs** を **1** に設定します。Type Properties ウィンドウで **Accept** をクリックし、変更を保存します。これによりアセット内に入力ノードが作成され、ネットワークに接続できます。後でこの入力を使用して、Unreal シーンからジオメトリを選択します。

Minimum Inputs は **0** のままにしておきます。これより高く設定すると、入力の最小要件が満たされないと、アセットが機能しません。そうなるとアセットはクックされず、何も起こりません。



02 ネットワークエディタで、grid の後に **Switch** ノードを追加し、新しい Input に接続します。**Subdivide** ノードを追加して入力ノードの後に接続し、アトリビュート値のランダム化の際に十分なディテールが得られるようにします。**Depth** を **5** に設定します。

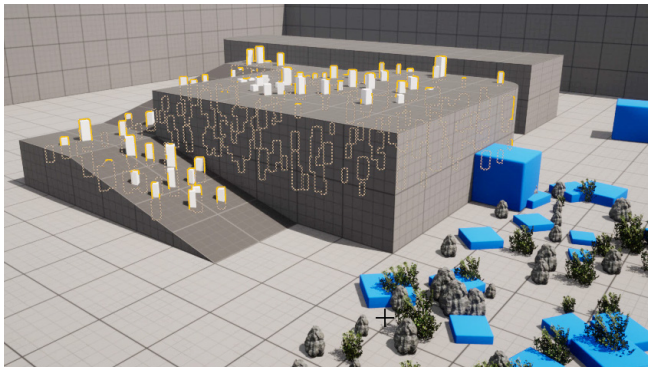
subdivide ノードに**エラー**が発生するのは、Houdini で 3 つ目の入力に何も接続されていないからです。1 つ上のレベルに戻り、**box** をアセットの入力に接続して、これがどう機能するかをプレビューしやすくします。Houdini でボックスは球に細分化されますが、最終シーンではこのようにはなりません。



03 **Assets > Edit Asset Properties > Populate** を選択します。***switch*** ノードから、**Select Input** パラメータをパラメータリストにドラッグします。**Select Input** パラメータを選択します。**Menu** タブをクリックして、**Use Menu** をオンにします。

0, Grid、次に **1, Input Geometry** を追加します。**Type Properties** ウィンドウで **Accept** をクリックし、変更を保存します。

これにより、アセットの UI にメニューが追加され、Unreal で使用できるようになります。



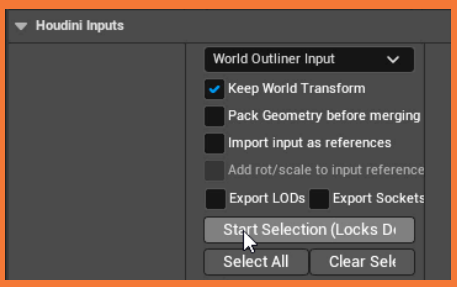
04 **UNREAL で** – **Rebuild** ボタンをクリックして変更を確定します。新しい ***populate*** アセットを前景にドラッグします。**Houdini Parameters** セクションに移動し、**Select Input** メニューから **Input Geometry** を選択します。

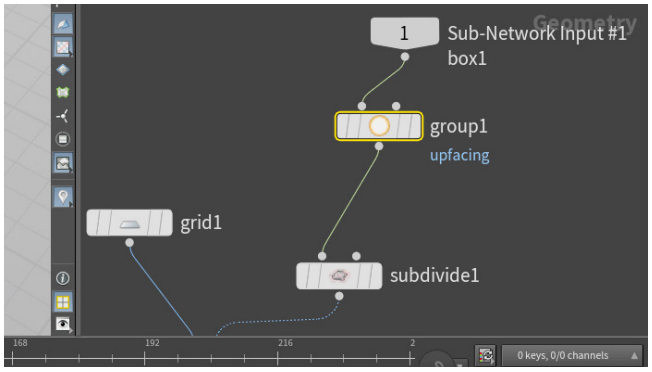
Details パネルで **Houdini Inputs** セクションに移動し、メニューから **World Outliner Input** を選択します。**Start Selection** ボタンをクリックし、3D シーンで傾斜とプラットフォームのすべてのパーツを選択します。**Use Current Selection** をクリックすると、インスタンスが細分化されたプラットフォーム内にばら撒かれます。しかし、私たちが求めているのはこれではありません。

UNREAL で入力ノードを使用する

アセットで入力ノードをセットアップする際、さまざまな方法で入力ジオメトリにアクセスできます。コンテンツブラウザからジオメトリを使用できます。

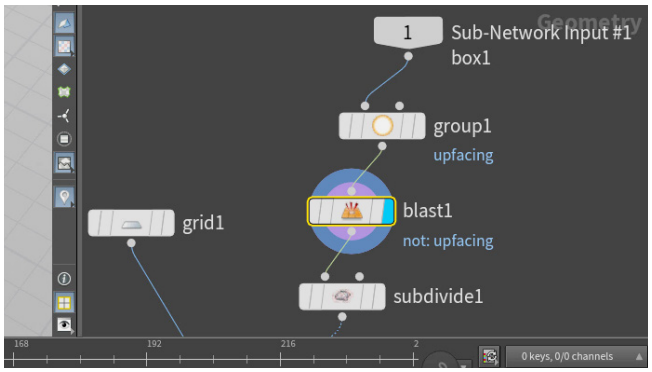
Curve Input をセットアップして、Unreal でカーブを描画できます。また、**World Outliner** からコンテンツを選択したり、Height Field に **Unreal のランドスケープ**を入力して、Houdini の新しい Terrain ツールセットで使用することもできます。





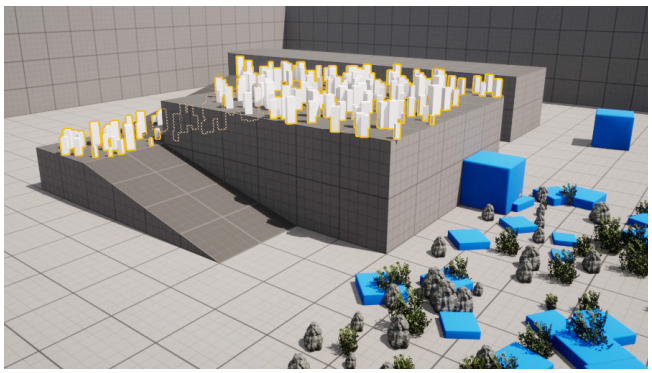
05 **HOUDINI で** – ボックスの上部のフェースを分離して、ポイントがコピーされる場所を制限します。ネットワークエディタで、***input*** ノードの後に **Group** ノードを挿入します。**Group Name** を **upfacing** に設定します。**Base Group** で、**Enable** を **オフ**にします。**Keep by Normals** で、**Enable** を**オン**にします。**Direction** を **0, 1, 0**、**Spread Angle** を **0** に設定します。

ポイントを Unreal 内のジオメトリのすべてのフェースにばら撒くのではなく、上面のフェースを分離してそれを使用するようにします。



06 ***group*** ノードの後に **Blast** ノードを挿入したら、**Group** を ***upfacing*** に設定し、**Delete Non Selected** を**オン**にします。これで、上を向いているプリミティブのみが維持され、その他は削除されます。**Assets > Save Asset > Populate** を選択します。

ここで重要なのは、グループを使用して上部のフェースを特定したので、ソリューションのアセットにどのジオメトリが入力されるのかは問題にならないところです。これが、プロシージャルアセット特有の動作です。一度限りのソリューションではなく、汎用のソリューションを提供します。



07 **UNREAL で – Rebuild Asset** ボタンをクリックして変更を確定します。これで、ジオメトリの上面にだけボックスが載っています。

このアセットで、デフォルトのグリッドまたは、レベルから取得したジオメトリを使用できるようになりました。Unreal で使用する Houdini デジタルアセットを構築するには、いくつもの方法があります。

08 **Number of Instances** を **40** に設定します。**Houdini Outputs** セクションに移動し、インスタンス化したボックスを展開します。

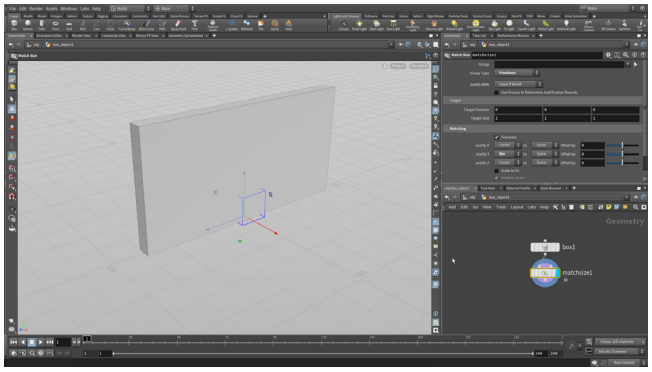
Content Browser で、**StarterContent > Particles** を開きます。***P_Fire*** プロップを **Houdini Instanced Input** 上にドラッグします。**Rotate Y** を **90** に設定します。**Play** を押して、レベルをテストします。

このアセットでインスタン化されたポイントは、ジオメトリ以外にも使用できます。これらのポイントは Unreal レベルの一部になっているので、さまざまな問題解決に利用可能です。シーンにさらにポイントを追加する場合は、炎を設定した ***populate2*** を非表示にしましょう。それでも、レベルを再生すると表示されます。

パート 7

Unreal に RBD シミュレーションをインポート

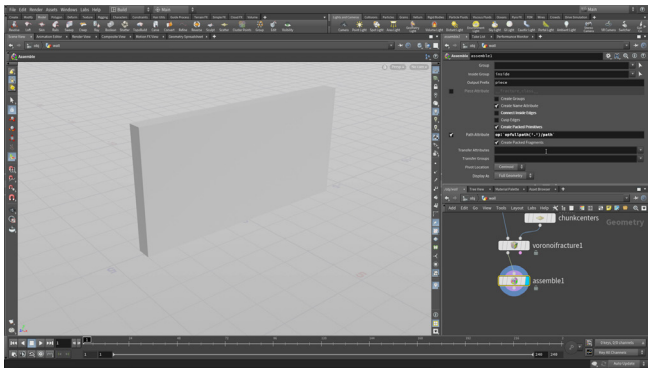
このパートでは、壁を作成し、リジッドボディダイナミクスを使用してそれを粉砕します。このシステムを FBX にエクスポートしたら、ゲームで使用するために Unreal にインポートします。Houdini から Unreal にビジュアルエフェクトを取り込む方法を、簡単な例で説明します。



01

HOUDINI で - **Box** を作成します。ジオメトリレベルに移動し、**Size** を **0.5, 4, 8** に設定します。

Match Size ノードを追加し、パラメータエディタの **Matching** で、**Justify Y** を **Min** にします。これでボックスが持ち上がり、地面の上に配置されます。



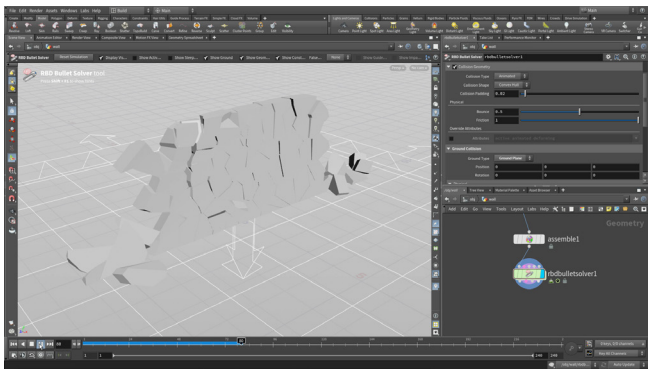
02

オブジェクトレベルに移動します。ノードの名前を ***wall*** に変更します。wall ノードを選択し、**Model** シェルフから **Shatter** を選択します。

ジオメトリレベルに入り、***chunkcenters*** ノードを選択し、**Force Total Count** を **100** に設定します。

チェーンの終端に Assemble ノードを追加します。**Connect Inside Edges** をオフにし、**Create Packed Primitives** と **Path Attribute** をオンにします。**Path Attribute** を次のように変更します。

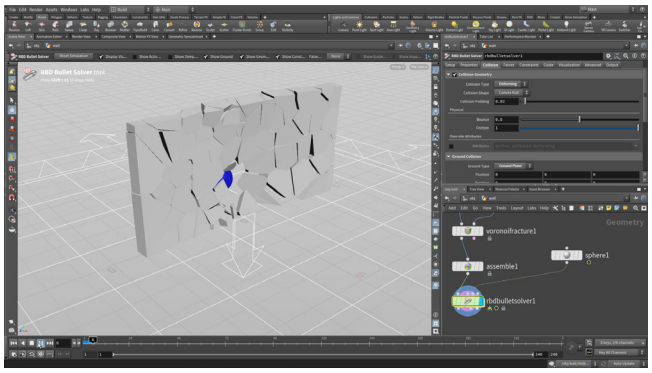
```
op:`opfullpath('.')/path'
```



03

チェーンの終端に **RBD Bullet Solver** ノードを追加します。**Ground Collision** タブで、**Ground Type** を **Ground Plane** に設定します。

Play を押して、シミュレーションを実行します。壁が砕けて落ちるだけで、特に面白みはありません。



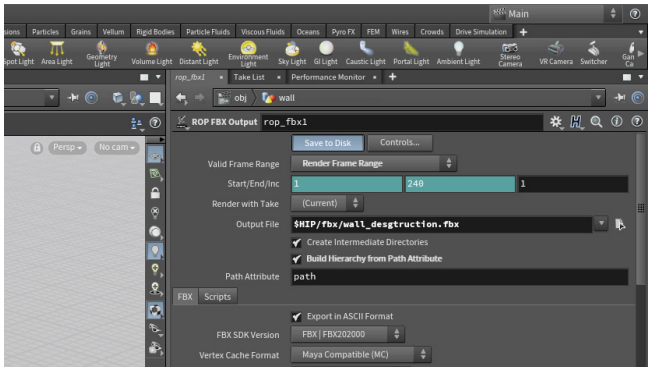
04

sphere ノードをネットワークに追加します。それを壁の前に置き、**Center X** を **2**、**Center Y** を **1** に設定します。**Alt** キーを押しながら **Center X** をクリックし、キーフレームを設定します。

フレーム 9 に移動します。**Center X** を **-3** に設定し、**Center X** を **Alt クリック**して 2 つ目のキーフレームを設定します。

sphere を ***rbdbulletsolver*** ノードの **4 つ目**の衝突入力に接続します。**Collisions** で、**Collision Type** を **Deforming** に設定します。

Play を押して、シミュレーションを実行します。球によって壁が砕かれるようになりました（シミュレーションでは、球は非表示になります）。



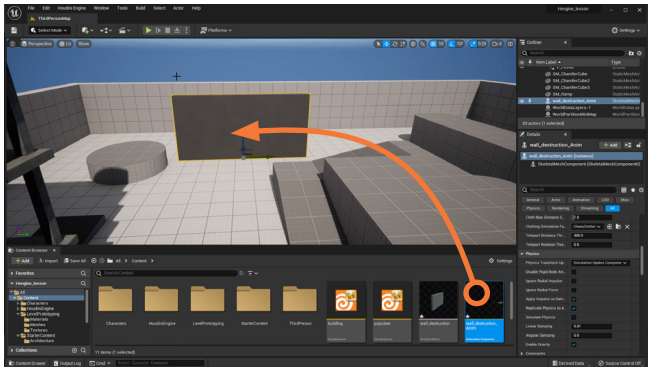
05

rbdbulletsolver ノードの後に **Transform** ノードを追加し、**Uniform Scale** を **100** に設定します。これで、シミュレーションが Unreal で必要なサイズにスケールされます。

FBX Output ノードを追加します。**Valid Frame Range** を **Render Frame Range** に、**Output File** を ***$HIP/fbx/wall_destruction.fbx*** に設定します。

Build Hierarchy from Path Attribute をオンにします。

Save to Disk をクリックして、ディスクに保存します。



06

UNREAL で – Content Browser で、***Content*** ディレクトリに戻ります。**Import** を押し、***wall_destruction.fbx*** ファイルを選択します。**Skeletal Mesh**、**Import Mesh**、**Import Animations** のチェックボックスをオンにします。

Animation セクションを展開し、**Import Meshes in Bone チェックボックスがオフになっていることを確認します。Mesh** セクションを展開し、**Normal Import Method** を **Import Normals and Tangents** に設定します。**Import** をクリックします。

UV がまだセットアップされていないというメッセージウィンドウが表示されます。これを閉じます。コンテンツリストに 4 つの新しいアイテムが表示されます。

wall_destruction_Anim アセットをワークスペースにドラッグします。

07

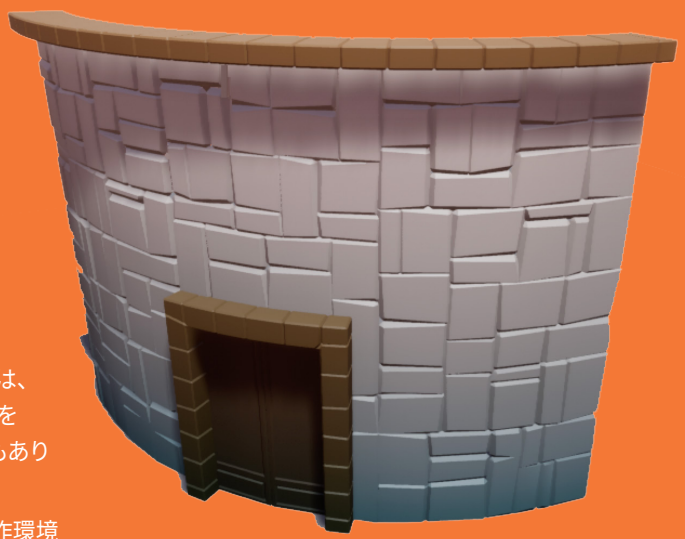
Play を押し、ゲーム内で動作しているシミュレーションを表示します。この時点ではアニメーションはループするだけで、相互作用もありません。ブループリントで、キャラクタの部位の動きに基づいてアニメーションをトリガするようにセットアップすることもできます。

まとめ

Unreal で使用する、さまざまなゲームアセットを作成しました。Houdini デジタルアセットからリジッドボディシミュレーションを含む FBX ファイルまで、基礎を習得しました。デジタルアセットを使用すると、Houdini のノードベースのワークフローを Unreal などのホストアプリケーションに統合できます。Unity、Autodesk Maya、Autodesk 3ds Max といった他のアプリケーションでも、同じワークフローを活用することが可能です。

詳細は、**SideFX.com/unreal** をご覧ください。ここでは、Unreal ですぐに使用できるアセットのスターターキットを入手できます。加えて、別のチュートリアルへのリンクもあります。

Project Titan のチェックもお忘れなく。これは 3D 制作環境を探求するためのインハウス技術デモで、Unreal の最新テクノロジーを活用しています。Project Titan 用に作成されたツールとテクニックは、学習資料およびダウンロード可能なコンテンツとして、コミュニティに共有されています。



HOUDINI FOUNDATIONS

PDG による都市構築

パイプラインのワークフローの管理には、Task Operator（TOP）を利用できます。TOP は PDG（Procedural Dependency Graph）と呼ばれるテクノロジを使って構築されています。PDG に基づいて作成されたワークフローは、TOP ノードを使用します。TOP ノードは、ローカルコンピュータまたは大規模な演算ファームにタスクを分散させるワークアイテムを生成します。

TOP ネットワークでは、各ワークアイテム間の依存関係や、各ワークアイテムが最終出力にどのように寄与するかを決められます。この情報はノードグラフで容易に視覚化でき、ネットワーク内でのデータフローを定義することができます。TOP を使えば、パイプラインの自動化、分析、スケールが可能なワークフローを構築できます。

このレッスンでは、TOP ノードを使って都市マップを取得し、都市ブロックごとにビルを作成した後、このシステムを拡張してより複雑なビルや大規模な都市マップを処理できるようにします。Houdini アーティストであれば、SOP でこれを行う方法をご存知でしょう。しかし、TOP を使えば、PDG のワークフローを学びながら、複数のタスクを外部の演算ファームに分散して並列処理させられる、スケールが容易なシステムを作成できます。

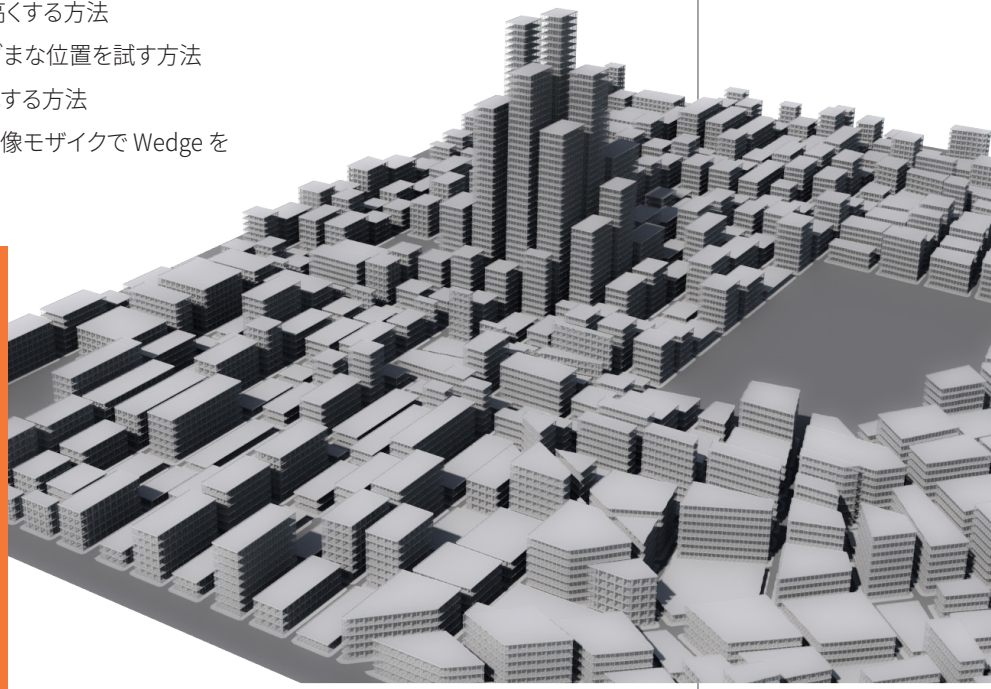
注：このレッスンでは、Image Magick を使用します。このアプリケーションがコンピュータにインストールされていることを確認してください。

レッスンの目標

- ***TOP（Task Operator）ネットワークを作成してプロシージャルな都市を構築し、それをレンダリングします。***

学習内容

- 都市マップ画像をジオメトリに変換する方法
- TOP ネットワークをセットアップして、都市ブロックを保存する方法
- TOP でジオメトリを作成し、各都市ブロックにビルを構築する方法
- 都心部を作成し、一部のビルを他よりも高くする方法
- 都市景観を Wedge 化し、都心部にさまざまな位置を試す方法
- さまざまなマップ画像の使用を Wedge 化する方法
- TOP を使用して都市をレンダリングし、画像モザイクで Wedge を比較する方法



使用する機能とソフトウェア

Houdini 19.5+ の機能を前提として、書かれています。

このレッスンの手順は、以下の Houdini 製品で実行可能です。

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

ドキュメントバージョン 4.0.1J | 2023 年 8 月

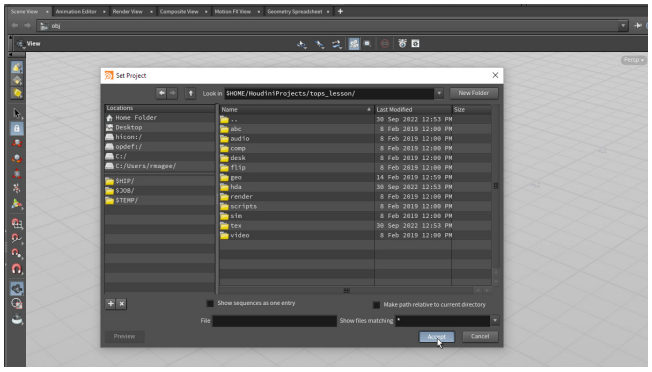
パート 1

都市グリッドの作成

プロシージャルな都市を構築するには、まず都市のグリッドを作成します。都市マップの白黒画像を含む画像ファイルをトレースすることで、ジオメトリを作成します。これが、TOP で構築するネットワークの入力ジオメトリとなります。

プロジェクトファイル

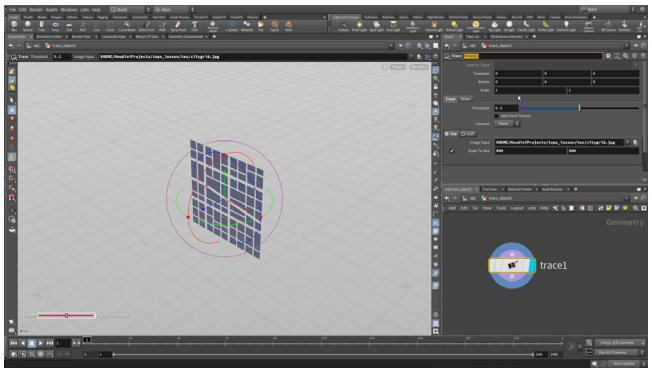
SideFX.com の **Foundations チュートリアル**のページ（このチュートリアルを入手した場所）から、***tops_lesson*** ディレクトリをダウンロードします。**home** または **documents** ディレクトリにある **Houdini Projects** ディレクトリに配置してください。



01

File > Set Project を選択します。ダウンロードした ***tops_lesson*** ディレクトリを見つけ、**Accept** を押します。これにより、先ほどコピーしたプロジェクトディレクトリとそのサブフォルダに、このショットに関連するファイルがすべて配置されるようになります。

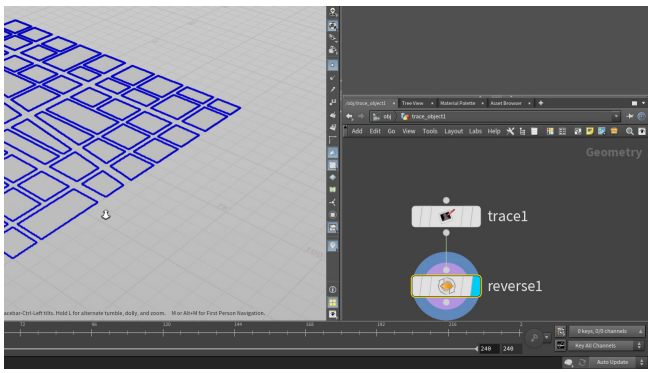
File > Save As... を選択します。新しい ***tops_lesson*** ディレクトリが表示されるはずです。表示されない場合は、左側の列で **$JOB** をクリックします。ファイル名を ***city_01.hip*** に設定し、**Accept** をクリックして保存します。



02

ビューポートで、**Tab を押して**メニューを開き、**TRACE** と入力していきます。**Trace** を選択すると、カーソルの位置に、シーン内への配置待ちの状態にある正方形の輪郭が表示されます。**Enter を押して**、原点の位置に配置します。現時点では、トレースされた円が表示されます。

ネットワークビューでノードを**ダブルクリック**して、ジオメトリレベルに入ります。***trace*** ノードを選択し、**Image Input** パラメータの横にある **File Chooser** をクリックします。**$HIP** をクリックしたら、***tex*** ディレクトリに移動します。***citygrid.jpg*** 画像を選択し、**Accept** をクリックします。**Scale to Size** オプションをオンにして、値を **500, 500** に設定します。これにより精度が向上します。



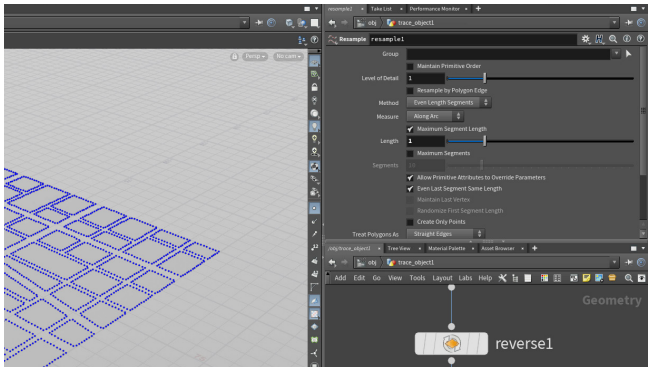
03

trace ノードの上部のセクションで、次のように設定します。

- **Rotate X** を **-90** にする
- **Scale** を **100, 100** にする

ネットワークビューで、**Tab** を押して **Reverse** と入力していきます。**Reverse** ノードを選択して配置したら、それに ***trace*** ノードを接続します。**Display フラグ**を設定します。このノードにより、法線が上を向くようになります。ビューポートで**スペースバー + H** を押して、都市グリッド全体を表示します。

ディスプレイオプションバーで **Display Points** をオンにすると、各都市ブロックの周りにトレースポイントが多数あることを確認できます。



04

ネットワークビューで **Tab > Resample** を押し、クリックして ***reverse*** ノードの下に配置します。まだ接続はしないでください。**Length** を **1** に設定したら、reverse ノードの出力を ***resample*** ノードの入力に接続します。

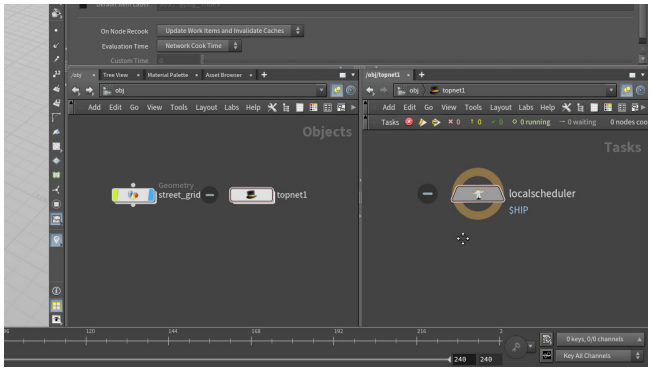
resample ノードの出力を **RMB クリック**し、**Null** を選択します。チェーンの終端に Null ノードをクリックして配置します。**Display フラグ**を設定し、名前を ***CITYBLOCKS_OUT*** に変更します。

作業内容を**保存**します。

パート 2

ワークアイテムの生成と表示

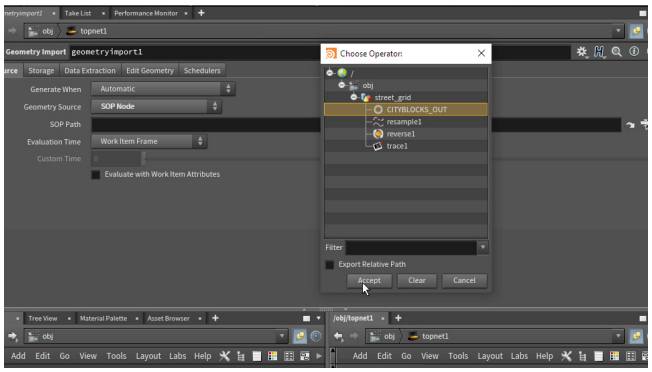
都市グリッドを作成できたら、さまざまな都市ブロックを個別のワークアイテムに分割します。これで、ブロックごとにビルを生成できるようになります。各ブロックを保存するための TOP ネットワークをセットアップします。同時に、選択したワークアイテムを視覚化し、望みどおりの結果になっていることを確認するためのオブジェクトも作成します。



01

オブジェクトレベルに戻り、オブジェクトの名前を ***street_grid*** に変更します。ネットワークビューで **Tab を押して**、**TOP...** と入力していき、**TOP Network** を選択します。クリックしてネットワークにノードを配置します。

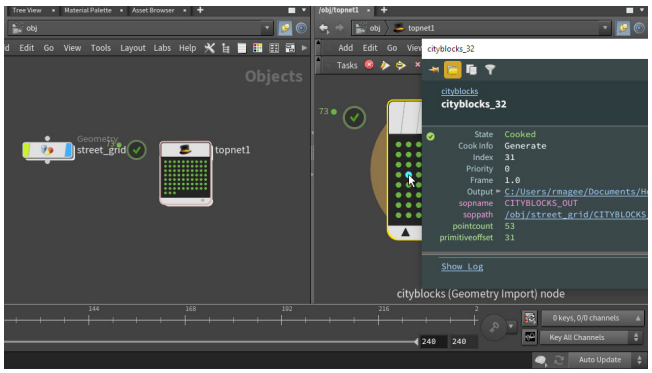
ネットワークビューの右上にある小さい**矢印**をクリックします。メニューから **Split Pane Left/Right** を選択します。左側のネットワークビューで**ピン**アイコンをクリックします。右側のネットワークビューはすでにピン留めされています。右側のネットワークビューで、***topnet*** を**ダブルクリック**して中に入ります。これで、両方のネットワークを使用して都市を構築できるようになりました。



02

TOP ネットワークで **Tab > Geometry Import** を押します。クリックしてノードを配置します。**Geometry Source** を **SOP Node** に設定したら、**SOP Path** の横にある **Choose Operator** アイコンをクリックします。フローティングウィンドウを使用し、***CITYBLOCKS_OUT*** ノードに移動して選択します。

Storage で、**Store Geometry As** は **External File** に設定したままにします。Data Extraction で、**Copy from Class** を **Primitive** に設定します。これにより、ファイルがディスクに格納されるようになるので、次の TOP ノードでそれらのファイルを取得できます。これは特に、演算ファームにタスクを分散する TOP ネットワークをセットアップする場合に重要となります。



03

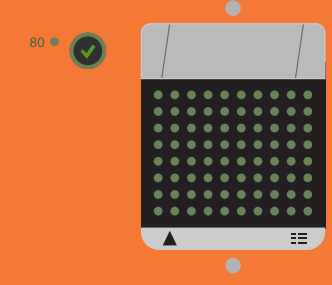
geometryimport ノードの名前を ***cityblocks*** に変更し、上部の**タスクバー**で **Cook Selected Node** ボタンをクリックします。または、このノードを選択して **Shift + G** を押しても、クックすることができます。ノードがクックされるとき、ワークアイテムを表す小さいドットが表示されます。

ドットを **Ctrl + MMB クリック**すると、ワークアイテムのアトリビュートが表示されます。**Index** などの基本の TOP アトリビュートは、すべてのワークアイテムに備わっています。また、ワークアイテムのタイプに固有のアトリビュートもあります。各ワークアイテムは、1 つの **Output** ファイルに関連づけられていることが分かります。ワークアイテムを **RMB クリック**して、**View Work Item Output** を選択すると、それに含まれるジオメトリを個別のジオメトリビューアで確認することができます。

ワークアイテム

TOP ノードは、実行を依頼されたタスクごとにワークアイテムを作成します。TOP ノード上で、これらはドットとして表現されます。これらのドットを使用して各ワークアイテムのステータスを視覚化したり、ドットを選択してそのワークアイテムの進捗状況を評価することができます。

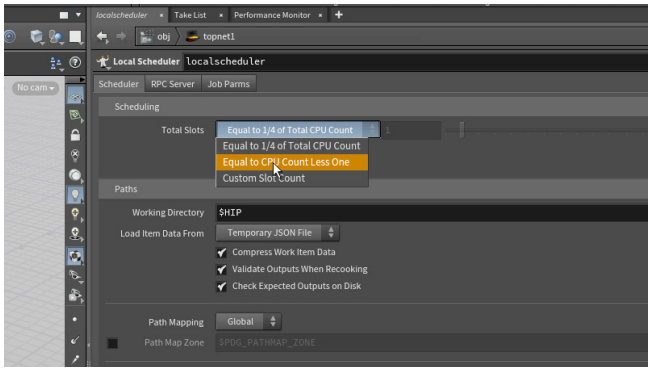
すべてのワークアイテムのクックが完了すると、ノードにチェックマークが付き、完了したワークアイテムが緑色になります。



スケジューラ系ノードとは

最初に TOP ネットワークを作成する際は、クック対象のタスクを管理する Local Scheduler ノードが作成されます。Local Scheduler はローカルコンピュータを指し、利用可能なコアの特定の割合をクックに使用します。**Total Slots** を **Equal to CPU Count Less One** オプションに設定すると、利用可能なコア数と同数になります。

また、より大規模な演算ファームにタスクを送信できるよう、**HQueue**、**Deadline**、**Tractor**、**Python** 用のスケジューラ系ノードをセットアップすることも可能です。これにより、並列に処理できるワークアイテムの数が増え、グラフの効率が向上します。

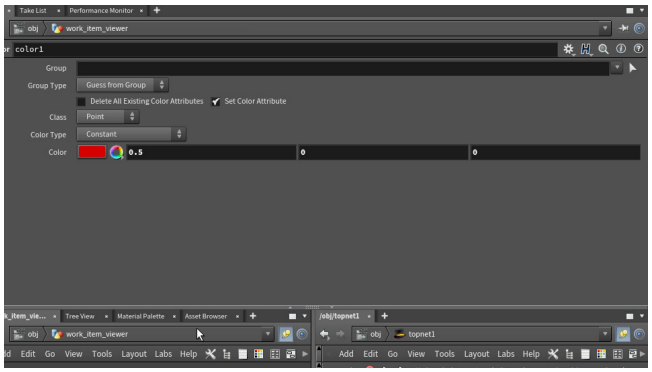


04

ネットワークでの処理速度を上げたい場合は、***localscheduler*** ノードを選択して、**Total Slots** を **Equal to CPU Count Less One** に設定します。使用するプロセッサが増えるため、以降のクックの速度が向上します。

ワークアイテムをビューポートに表示させるには、ワークアイテムビューアをセットアップする必要があります。左側のネットワークビューで、**Tab > File** を押します。クリックしてノードを配置し、名前を ***work_item_viewer*** に変更します。

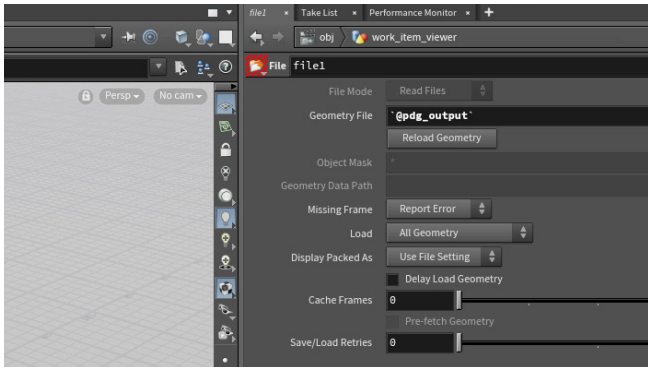
File ノードは通常、ディスク上のファイルを指します。まずはこの方法でファイルを見つけ、その後、別の方法を使ってワークアイテムを TOP から直接取得します。



05

work_item_viewer ノードをダブルクリックしてその中に入り、パラメータエディタで **Geometry File** ファイル選択ボタンをクリックします。ファイル選択ダイアログで、**$HIP** をクリックしてから ***geo*** フォルダを**ダブルクリック**します。

city_01_cityblocks ファイルシーケンスを選択します。保存されているジオメトリが、番号付きのシーケンスとしてインポートされます。ジオメトリシーケンスの後に **color** ノードを追加し、そのカラーを**赤色 (1, 0, 0)** に設定します。タイムラインをスクラブすると、さまざまなジオメトリが順番にロードされる様子を確認できます。ディスクに保存されたジオメトリは 73 個です。

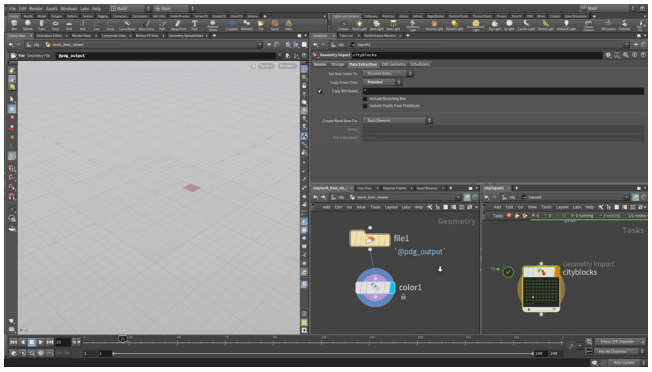


06

都市ブロックの表示をフレーム番号にリンクするのではなく、TOP ネットワークに直接リンクしてみましょう。**File** ノードをクリックして、**Geometry File** を次のエクスプレッションに変更にします。

```
`@pdg_output`
```

これは、ディスクから順番にファイルをロードするのではなく、topnet でアクティブになっているワークアイテムをロードすることを意味します。PDG ネットワークから出力されるワークアイテムがないため、最初はエラーが返されます。



07

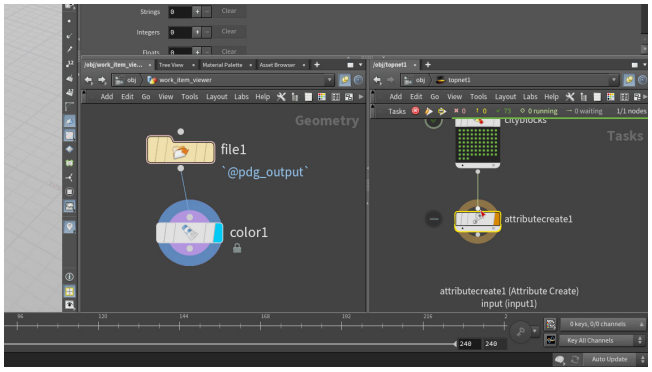
ジオメトリネットワークビューの **Active Work Item** メニューで、異なるワークアイテムを選択し、Scene View でそれらを選択します。赤色が選択したジオメトリに移動します。そのワークアイテムに関連するジオメトリがビューポートに表示されます。topnet では、黄色にハイライトされた ***cityblocks*** TOP ノード上にドットが表示されます。

作業内容を**保存**します。

パート 3

アトリビュートの追加

ビルを作成するには、固定のベース高さと、ランダムな高さのバリエーションを設定して、ビルによってサイズが異なるようにする必要があります。これらのアトリビュートは、都市マップのジオメトリレベルでセットアップできますが、ここ（TOP）で割り当てることも可能です。そうすれば、後で TOP レベルで変更を加える必要が生じても、容易に対処できます。



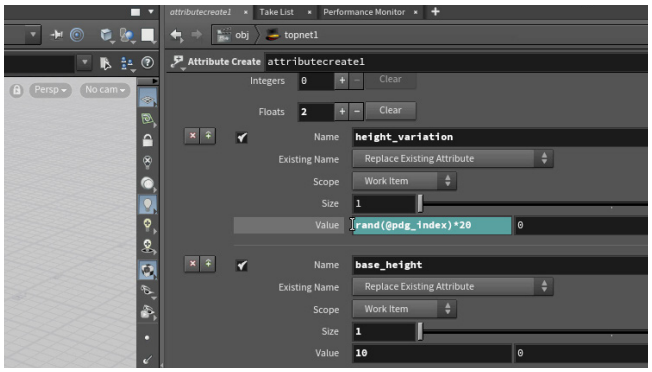
01

TOP ネットワークで、**Attribute Create** TOP を追加します。**Generate When** を **Each Upstream Item is Cooked** に設定します。

Float Attributes の下の **+（プラス）** 記号をクリックして新しいアトリビュートを作成し、次のように設定します。

- **Name** を ***base_height*** にする
- **Value** を **10** にする

これにより、すべてのビルが 10 の最小値を持つことになります。最も低いビルの高さを変えたい場合は、後でこの値を調整します。

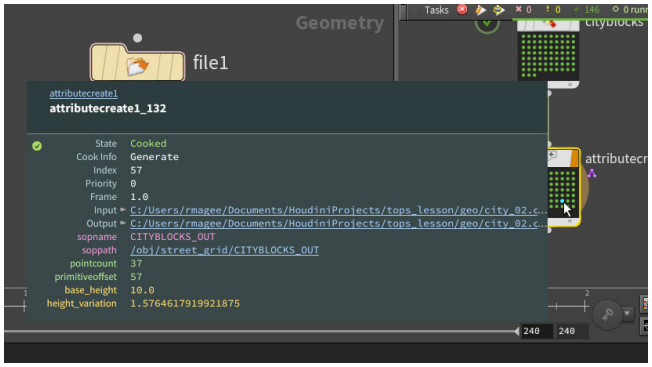


02

+（プラス） 記号を再度クリックし、次のように設定します。

- Name を ***height_variation*** にする
- **Value** を `rand(@pdg_index)*20` にする

この場合、ワークアイテムのアトリビュート **Index** をシードとして使用して、0 から 20 までの乱数が作成されます。ビル作成用のネットワークを構築する際は、この値をベース高さの値に加算して、ビルの高さが決定されます。



03

attributecreate ノードを選択し、**Shift + G** を押してクックします。ビルをまだ作成していないため、3D ビューには依然としてフットプリントが表示されます。

ワークアイテムを **MMB クリック**すると、それぞれのワークアイテムに値 **10** の ***base_height*** と、**0 と 20** の間の数値である ***height_variation*** が含まれていることが分かります。アトリビュートはこの TOP ノードで生成されましたが、まだ使用されていません。

TOP でアトリビュートを追加する

都市グリッドのジオメトリネットワークでアトリビュートを追加することもできましたが、TOP でアトリビュートを追加すれば、TOP のコンテキスト内で簡単に変更を加えられます。

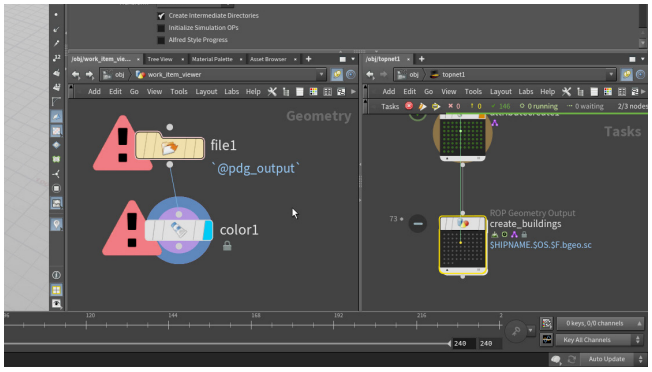
アトリビュートは、パイプラインに対して重要な情報を提供する重要な手段であるため、適切にセットアップすることが大切です。

```
pointcount       29
primitiveoffset  26
base_height      10.0
height_variation 19.69480037689209
```

パート 4

都市グリッド用のビルの作成

ROP Geometry ノードを使用して、シンプルなビルを作成します。作業内容をビューポートで確認するには、ワークアイテムを生成し、そのいずれかを選択する必要があります。その後、そのワークアイテムを使用して、ジオメトリレベルでビルをデザインします。

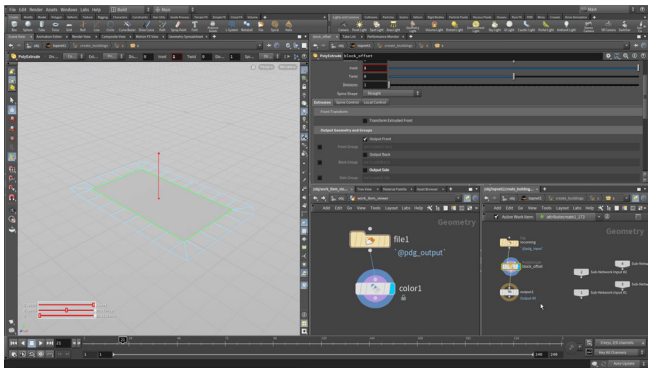


01

ROP Geometry Output TOP を追加します。名前を ***create_buildings*** に変更します。**Use External SOP** オプションを**オフ**にします。

ノードを **RMB クリック**して **Generate Node** を選択し、ワークアイテムを作成します。これらは、まだ処理されていない灰色のワークアイテムです。これらは実質上、ノードのセットアップ後に実行されるタスクのためのプレースホルダーです。

重要：ワークアイテムの 1 つをクリックしてみてください。このノードではまだジオメトリが生成されていないため、***work_item_viewer*** ネットワークでエラーが生成されます。

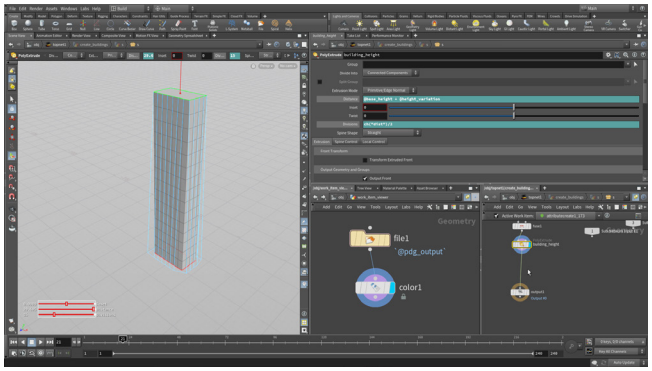


02

このノードを**ダブルクリック**して、ジオメトリレベルに入ります。

PolyExtrude ノードを作成し、***incoming*** と ***output*** ノードの間に配置します。名前を ***block_offset*** に変更します。**Inset** を 1 に設定し、**Extrusion** で **Output Side** オプションを**オフ**にします。これで、ビルをインセットして歩道を作成できるようになります。

次に、***polyextrude*** ノードの後に **Fuse** ノードを追加し、**Snap Distance** を **0.2** に設定して、小さい線をすべて除去します。***work_item_viewer*** ノードでエラーが発生しているにもかかわらず、ビューポートに何かが表示されていることに注意してください。この問題は後で修正します。



03

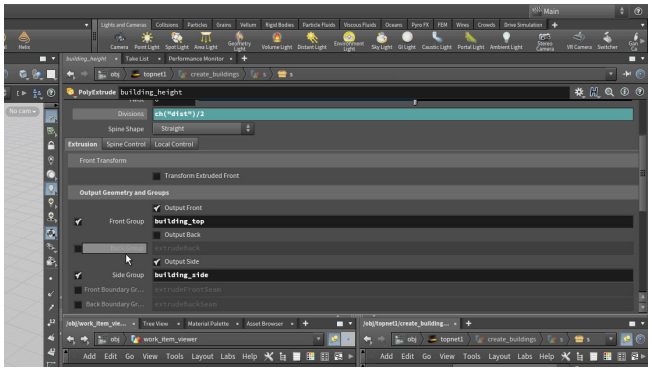
fuse の後に、2 個目の **PolyExtrude** ノードを作成します。名前を ***building_height*** に変更し、**Distance** を次のように設定します。

```
@base_height + @height_variation
```

Distance パラメータがハイライトされていることを確認してください。パラメータを **RMB クリック**し、**Copy Parameter** を選択します。**Divisions** パラメータを **RMB クリック**し、**Paste Relative References** を選択します。次のように 2 で割って、結果のチャネル参照を編集します。

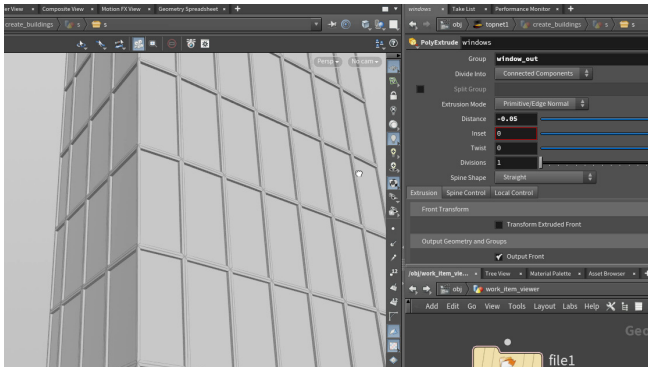
```
ch("dist")/2
```

これで、床と窓のグリッドが表示されるようになります。



04

Extrusion タブで、**Front Group** を**オン**にし、名前を ***building_top*** に変更したら、**Side Group** を**オン**にして名前を ***building_side*** に変更します。これらを使用してビルにディテールを追加していきます。

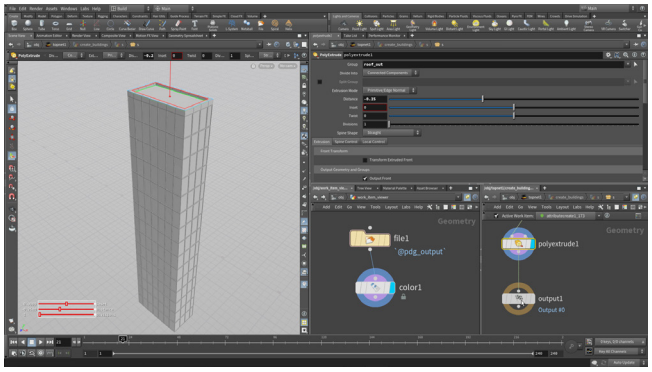


05

Poly Extrude ノードをチェーンに追加し、名前を ***window_frames*** に変更します。次のように設定します。

- **Group** を ***building_side*** にする
- **Inset** を 0.05 にする
- **Divide Into** を **Individual Elements** にする
- **Extrusion** タブで、**Front Group** を**オン**にして、その名前を ***window_out*** に変更します。

Poly Extrude ノードをもう 1 つ追加して、名前を ***windows*** に変更します。**Group** を ***window_out*** に、**Distance** を **-0.05** に設定します。



06

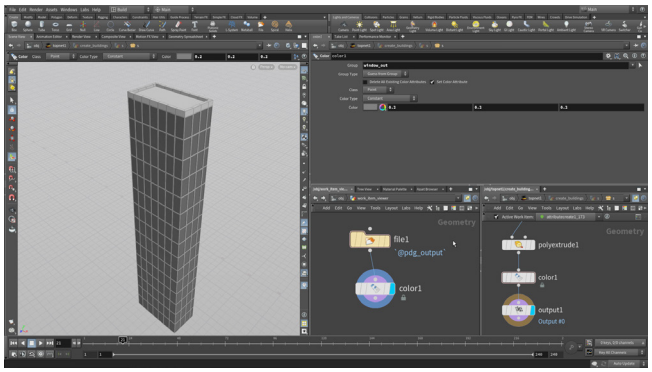
新しい **Poly Extrude** を追加し、名前を ***extend_roof*** に変更します。次のように設定します。

- **Group** を ***building_top*** にする
- Distance を **0.5** にする

Poly Extrude ノードをもう 1 つ追加して、名前を ***roof_edge*** に変更します。次のように設定します。

- **Group** を ***building_top*** にする
- Inset を **0.3** にする
- **Extrusion** タブで、**Front Group** を**オン**にして、その名前を ***roof_out*** に変更します。

新しい **Poly Extrude** ノードをチェーンに追加し、名前を ***roof*** に変更します。**Group** を ***roof_out*** に、**Distance** を **-0.25** に設定します。

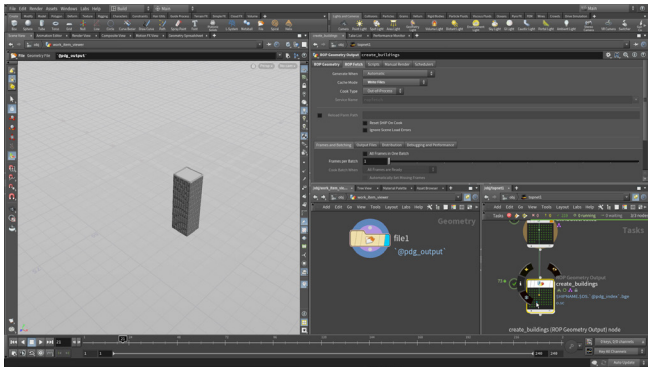


07

この後に color ノードを追加し、次のように設定します。

- **Group** を ***windows_out*** にする
- **Color** を**ダークグレー (0.2, 0.2, 0.2)** にする

これらのノードが ***output*** ノードに接続されていることを確認し、そのノードの **Display フラグ**を設定します。これで、枠が明るい色に、窓が暗めの色になり、ビューポートで識別しやすくなります。



08

1 つ上のレベルに戻り、***create_buildings*** TOP で `$F` を `` `@pdg_index` `` に変更し、**Output File** を次のように設定します。

```
$HIP/geo/$HIPNAME.$OS.`@pdg_index`.bgeo.sc
```

これにより、フレーム番号の代わりに各ビルのインデックスファイルが使用されるようになります。**ROP Fetch** タブをクリックし、**Cache Mode** を **Write Files** に設定します。

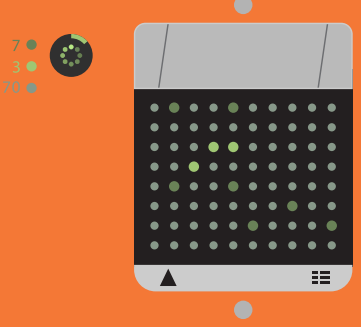
シーンファイルを**保存**したら、***create_buildings*** TOP を選択し、**Shift + G** を押してクックします。

work_item_viewer 内に戻り、赤色が表示されないように color ノードを削除します。***create_buildings*** TOP ノードで、ワークアイテムをクリックすると、さまざまな高さのビルが表示されます。

ワークアイテムの進捗状況

これまで、ワークアイテムは非常に素早く処理されてきましたが、このノードはやや長い時間を要します。進捗状況のホイールは、完了したタスクを暗い緑色で、進行中のタスクを黄色で、キュー内のワークアイテムを灰色で表示します。これを使えば、割り当てられたタスクがノード上で実行されていく様子を確認できます。

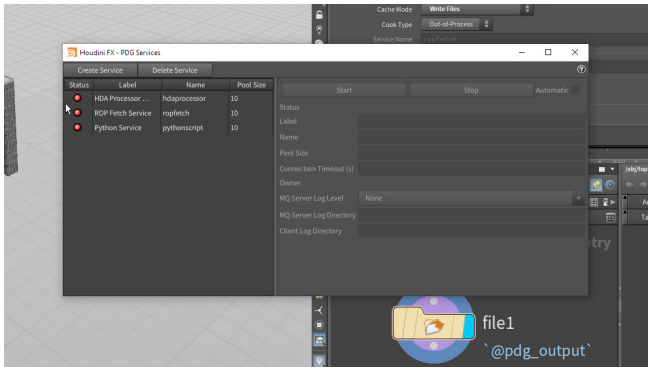
オブジェクトレベルの TOP ネットワークにも、ネットワークのすべてのタスクについてワークアイテムの進捗状況が表示されます。



パート 5

ビルの結合

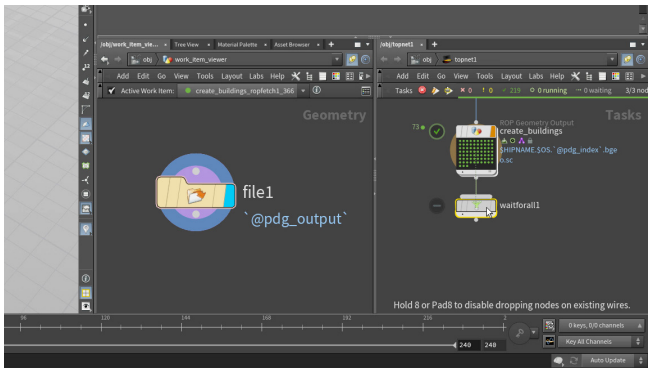
ビルが完成したら、それらのビルを再度まとめて都市にします。そのためには、Wait for All と呼ばれるパーティションノードと、Geometry Import を使用します。また、他の場所よりも高いビルが立つ都心部を追加します。



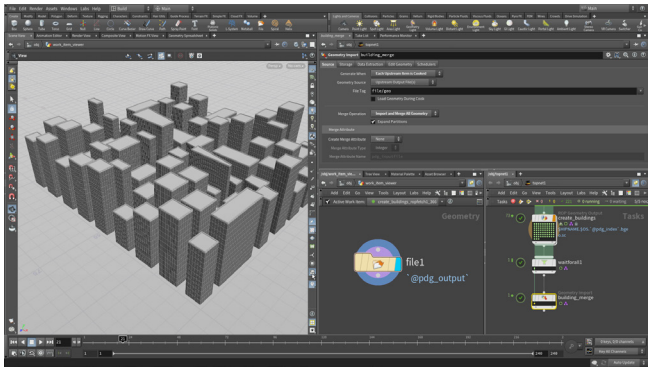
01 タスクの処理速度を上げるには、PDG Services というものを使用します。TOP ネットワークビューで **Tasks > PDG Services** を選択します。ここではサービスがセットアップされ、使用できるようになっていることが確認できます。

Services ウィンドウを閉じたら、***create_buildings*** TOP の **ROP Fetch** タブに移動して、**Cook Type** を **Services** に設定します。これで TOP ノードの処理速度が向上するはずです。

シーンファイルを**保存**したら、***create_buildings*** TOP を選択します。**Shift + D** を押して Dirty（変更あり）にしてから、**Shift + G** を押してクックします。先ほどよりもずっと速くなっているはずです。

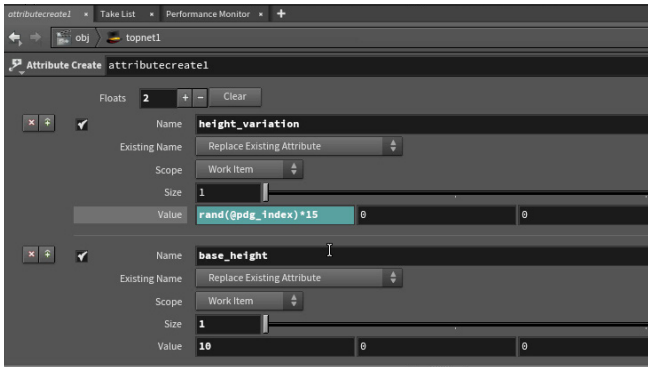


02 **Wait for All** TOP ノードを追加します。このノードは、ROP Geometry ノードによって生成されたすべてのワークアイテムを受け取り、それらを結合して単一のワークアイテムにします。このノードはまた、上流のすべてのワークアイテムのクックが完了するまで、下流の TOP がクックされないようにします。



03 **Geometry Import** TOP を追加します。名前を ***building_merge*** に変更します。**Generate When** を **Each Upstream Item is Cooked** に、**Merge Operation** を **Import and Merge All Geometry** に設定します。

シーンファイルを**保存**したら、***building_merge*** ノードを選択し、**Shift + G** を押してクックします。ネットワークのクックが完了したら、最終的な ***building_merge*** ノードのワークアイテムをクリックします。ビューポートに都市全体が表示されます。

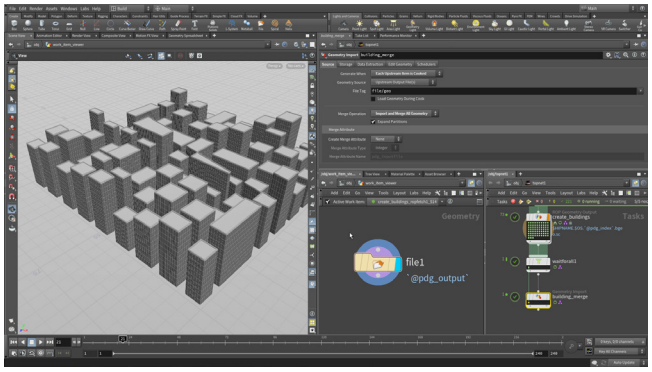


04 **Attribute Create TOP** に戻り、***height_variation*** アトリビュートを次のように編集します。

- **Value** を `rand(@pdg_index) * 15` にする

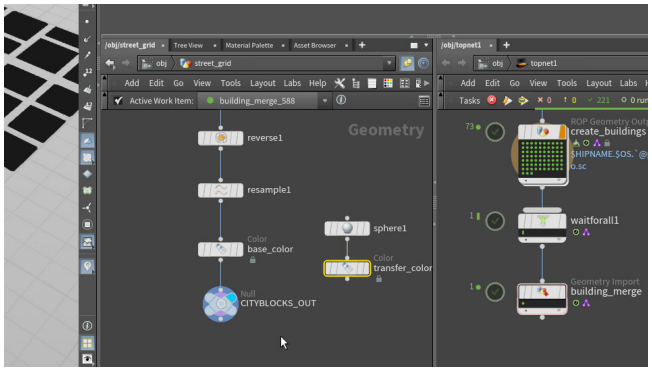
これで、ビルの高さが全体的に少し低くなります

attributecreate TOP を **RMB クリック**し、**Dirty This Node** を選択します。これにより、チェーン内の以降のワークアイテムがすべてクリアされます。



05

シーンファイルを**保存**します。***building_merge*** TOP ノードを選択し、**Shift + G** を押してクックします。ネットワークのクックが完了したら、最終的な ***building_merge*** ノードのワークアイテムをクリックして変更を確認します。



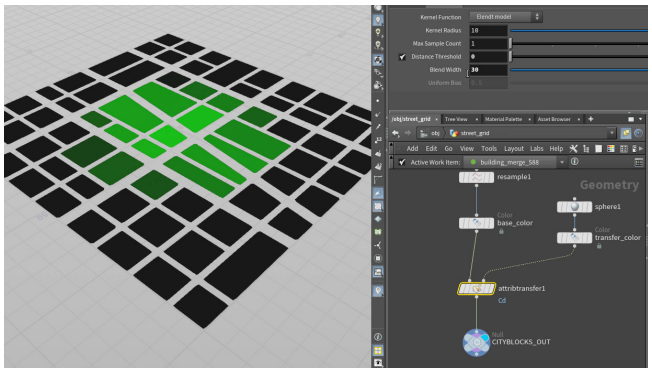
06

street_grid オブジェクトに移動し、その中に入ります。**color** ノードを作成し、チェーンの ***resample*** ノードの後に接続します。**Class** を **Primitive** に、**Color** を **黒色 (0, 0, 0)** に設定します。このノードの名前を ***base_color*** に変更します。

street_grid ネットワークに球を作成し、端の方に配置して次のように設定します。

- **Primitive Type** を **Polygon** にする
- **Uniform Scale** を **5** にする

sphere の出力に Color ノードを追加し、**Class** を **Primitive** に、**Color** を **緑色 (0, 1, 0)** に設定します。このノードの名前を ***transfer_color*** に変更します。



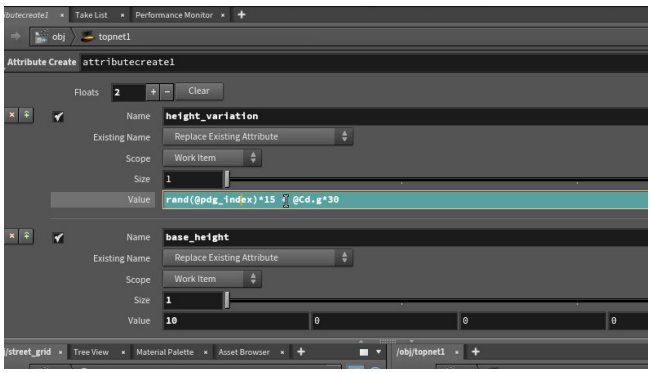
07

base_color ノードの後に **Attribute Transfer** ノードを追加します。2 つ目の入力に ***transfer_color*** ノードを接続します。

Points チェックボックスをオフにして、**Primitives** フィールドを **Cd** に設定します。**Conditions** タブで、次のように設定します。

- **Distance Threshold** を **0** にする
- **Blend Width** を約 **30** にする

カーソルを Scene View に置いた状態で、**スペースバー + Y** を押し、**Hide Other Objects** に切り替えます。タイルが球の中心に近づくにつれ、緑色が徐々にタイルに浸透していくのが分かります。



08

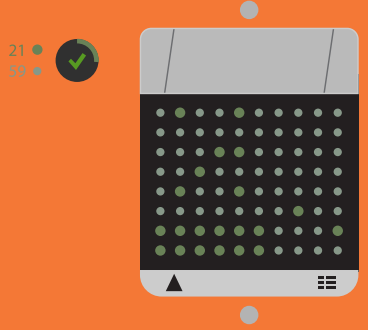
TOP ネットワークに戻り、***attributecreate*** を選択して、***height_variation*** アトリビュートを次のように編集します。

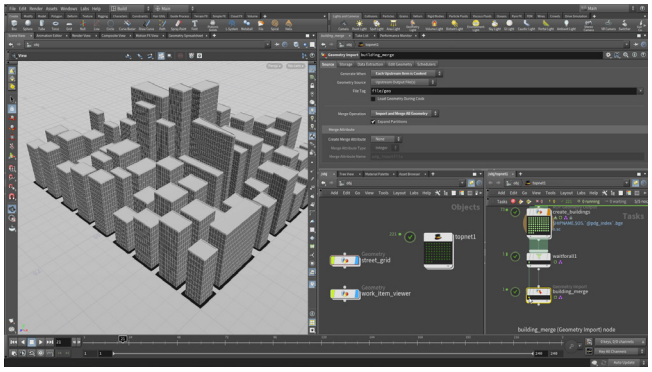
- **Value** を `rand(@pdg_index) * 15 + @Cd.g*30` にする

これにより、都市部のビルの高さが高くなります。最大で、値 30 だけ高さが追加されるようになります。ビルの高さを低くまたは高くしたい場合は、この値を変更します。

Dirty (変更あり) とクリーンなワークアイテム

PDG テクノロジーの主な目的の 1 つは、複雑なシステムに生じる依存関係です。都心部に変更を加えると、Dirty (変更あり) になって再クックが必要となるノードと、そのままで問題のないノードに分かれます。TOP におけるこうした依存関係の処理方法が、TOP の強みの 1 つです。

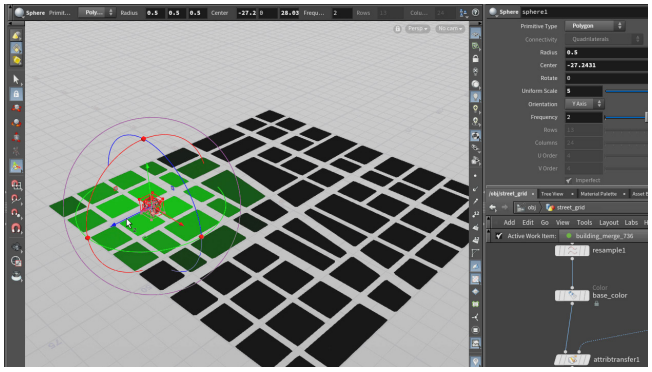




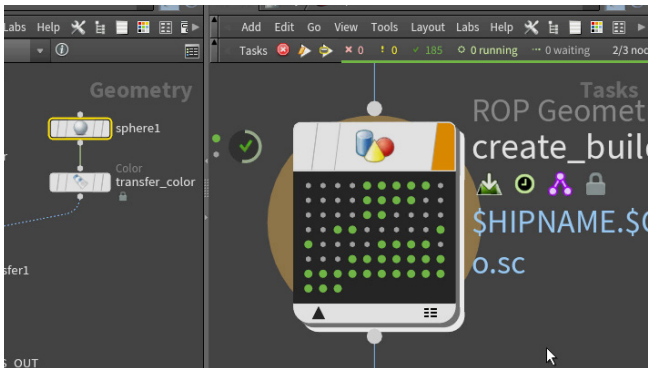
09 ***building_merge*** ノードを **RMB クリック**し、メニューから **Dirty This Node** を選択します。または、このノードを選択して **Shift+ D** を押しても、Dirty（変更あり）にできます。

building_merge ノードを選択し、**Shift + G** を押してクックします。保存するかどうかを確認されます。準備が整ったら、最終的な ***building_merge*** ノードのワークアイテムをクリックします。

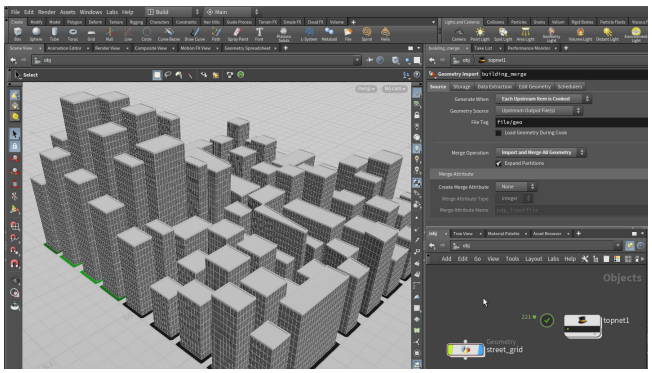
もう一方のネットワークビューで、オブジェクトレベルに戻ります。都市が更新され、ビューポートで都心部を確認できるはずです。



10 street_grid オブジェクトに移動し、***CITYBLOCK_OUT*** ノードを表示した状態で、sphere ノードを選択します。**ハンドル**ツールを使ってこのノードをあちこち動かすと、都市グリッドのさまざまな部分に作用しているのが分かります。



11 TOP ネットワークに戻り、***create_buildings*** ノードを **RMB クリック**して **Generate Node** を選択します。ジオメトリレベルで加えた変更のため、一部のワークアイテムが自動的に Dirty（変更あり）にされているのが確認できます。その他のワークアイテムはまだクリーンとみなされるため、再クックの必要はありません。

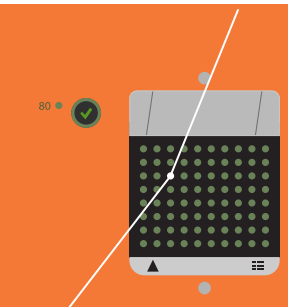


12 ***building_merge*** ノードを **RMB クリック**し、**Cook Selected Node** を選択すると、Dirty（変更あり）のワークアイテムのみが更新されます。ワークアイテムをクリックして、新しい都心部を表示します。このようにすると、ワークフローを開発する際、TOP の依存関係グラフを効率的に動作させることができます。

依存関係の更新

あるワークアイテムを選択すると、そのワークアイテムがシステム内の他のワークアイテムとどのように接続しているかを示す線が表示されます。

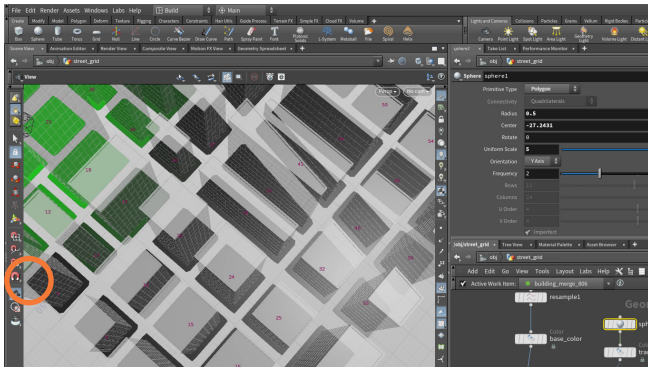
これは依存関係のマッピングであり、グラフを効率的に動作させるのに役立ちます。なぜなら、**Shift + D** でノード上のすべてのワークアイテムを明示的に Dirty（変更あり）にしない限り、Dirty（変更あり）のワークアイテムのみが処理されるからです



パート 6

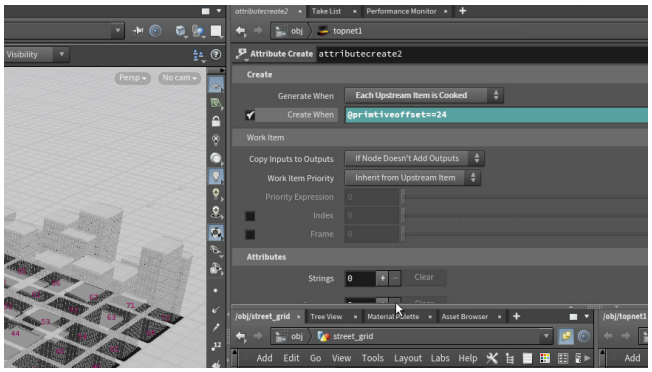
1 つのビルを分離する

これまで、ビルの高さは、height_variation のランダム性によって決定されてきました。あるビルを特定の高さに固定したい場合は、2 つ目の attribute create ノードを使用してプリミティブを選択し、特定の値に設定します。この手法を使うと、アーティスティックな指示をシステムに簡単に出すことができ、ランダム性を上書きできます。



01 **スペースバー + Y** を押し、**Ghost Other Objects** に切り替えます。***street_grid*** オブジェクトに戻り、**ディスプレイオプション** の **Display Primitive Numbers** をオンにします。ブロック番号が表示されます。これで、ビルを特定し、そのいずれかを明示的に設定するかどうかを決められるようになりました。ここでは**ブロック 24** を選択します。

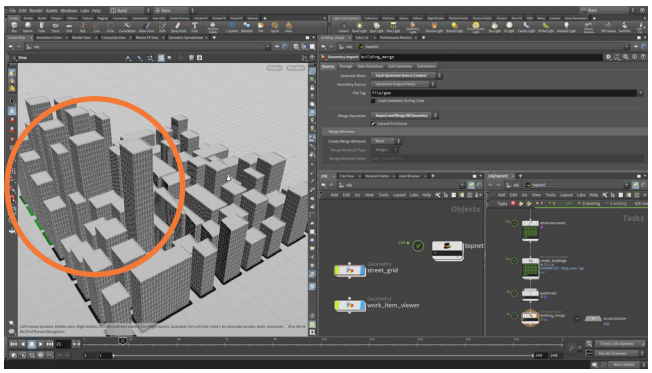
現在のセットアップのランダム性に依存する代わりに、この都市ブロックに特定のパラメータを設定して、望みどおりの正確な高さが得られるようにしましょう。すべてをランダムに任せずに、TOP ネットワーク内でアーティスティックな制御ができるようになります。



02 attributecreate TOP ノードを **Alt + クリック**しながらドラッグして、そのコピーを作成します。まだ接続はしないでください。**Create When** の横にあるチェックボックスをオンにし、次のエクスプレッションを追加します。

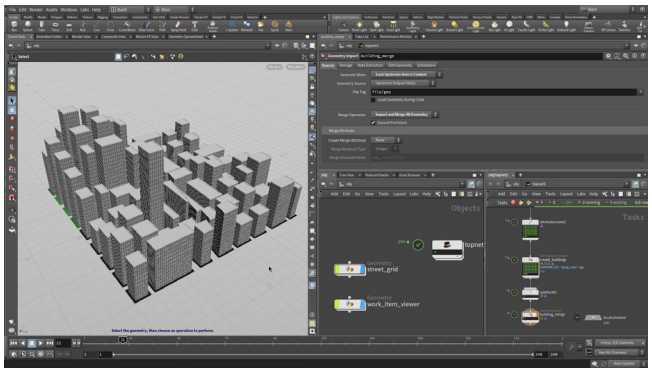
@primitiveoffset==24

height_variation の **Value** パラメータを **RMB クリック**し、**Delete Channels** を選択してから、その値を **0**、***base_height*** の値を **50** に設定します。これで、**ブロック 24** のビルの高さを明示的に 50 単位に設定できます。



03 このノードを、***attributecreate*** と ***create_buildings*** ノードの間に挿入します。作業内容を**保存**してから、***building_merge*** ノードを再クックします。

create_buildings ノードのインデックス 24 のワークアイテムをクリックすると、ブロック 24 に 50 単位の高さのビルが表示されます。



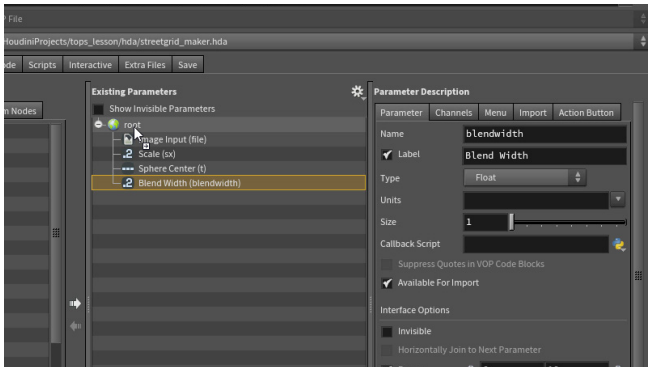
04 ***building_merge*** ノードのワークアイテムをクリックして、新しいビルがそびえ立つ都市景観を表示します。これで、ネットワークを再クックしてもブロック 24 のビルは更新されず、明示的に設定された状態に保たれるようになります。

そのビルの高さを変更したい場合は、2 つ目の Attribute Create ノードを使用して、その高さを明示的に設定します。

パート 7

都心部の位置を Wedge 化する

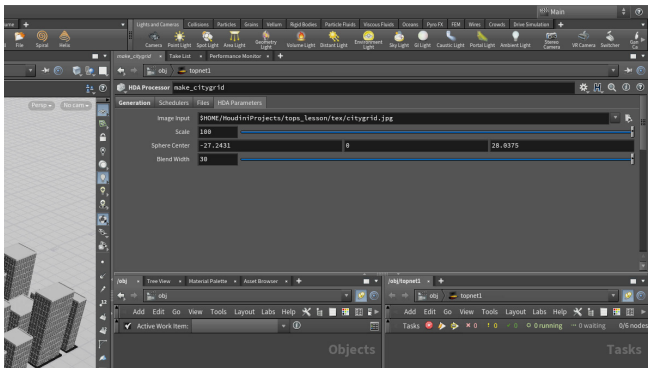
street_grid オブジェクト内で球の位置を Wedge 化するには、ネットワークを Houdini Digital Asset（HDA）に変換し、HDA Processor TOP ノードを介して実行する必要があります。これらのアセットをロードしてシステムに追加できるようになると、個別のツールから複雑なシステムを作成し、ツールを必要に応じて更新および適応することができます。



01

street_grid ネットワークで、***CITYBLOCKS_OUT*** を除くすべてのノードを選択します。**Assets** メニューから **New Digital Asset from Selection** を選択します。アセットに ***streetgrid_maker*** という名前を付け、ラベルを ***Street Grid Maker*** に設定したら、`$HIP/hda/` ディレクトリに保存します。

Operator Type Properties パネルで **Parameters** タブをクリックします。そのアセットの中に入ります。***trace*** ノードからパラメータリストに **Image Input** をドラッグします。**Scale X** を **Scale Y** にドラッグして、**Relative Channel Reference** を選択します。次に **Scale X** をパラメータリストにドラッグして、名前を ***Scale*** に変更します。***sphere*** ノードから **Center** をドラッグして、名前を ***Sphere Center*** に変更します。***attributetransfer*** ノードから **Blend Width** をドラッグし、**Accept** をクリックします。

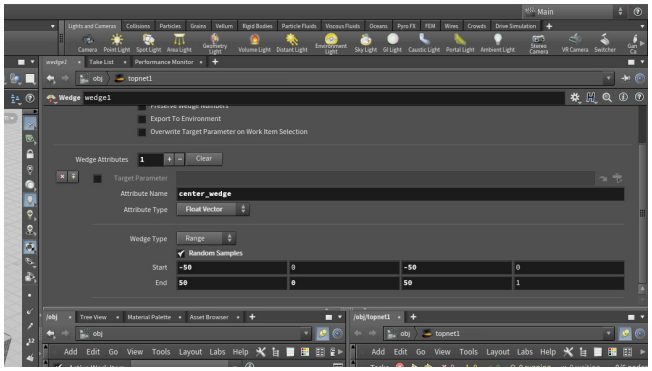


02

TOP ネットワークに戻ります。**HDA Processor TOP** を作成し、それを **Geometry Import** ノードに接続します。***geometryimport*** ノードで、**Generate When** を **Each Upstream Item is Cooked** に、**Geometry Source** を **Upstream Output File** に変更します。

HDA Processor ノードを選択し、**HDA File** を ***$HIP/hda/streetgrid_maker.hda*** に設定します。**HDA Parameters** タブをクリッ し、**Image Input**、**Center**、**Scale**、**Blend Width** パラメータを表示します。

この TOP の名前を ***make_citygrid*** に変更します。**Shift + V** を押し、ノードを ***Dirty（変更あり）にしてクック***し、このノードが以前のような方法で都市グリッドを生成していることを確認します。



03

Wedge TOP ノードを作成し、***make_citygrid*** に接続します。次のように設定します。

- **Wedge Count** を **4** にする

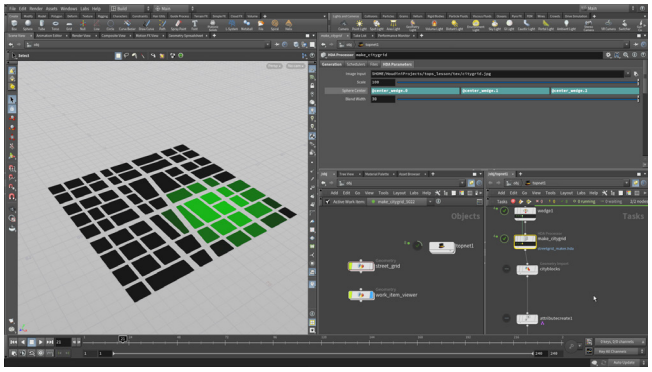
Wedge Attributes の横にある **+（プラス）記号**をクリックし、最初のアトリビュートの **Attribute Name** を ***center_wedge*** に設定したら、次のように設定します。

- **Type** を **Float Vector** にする
- **Start（範囲）**を **-50, 0, -50, 0** にする
- **End（範囲）**を **50, 0, 50, 0** にする
- **Random Samples** を**オン**にする

WEDGING

Wedging は、さまざまな設定で撮影しておいて、現像所に持ち帰って最良のものを選ぶという、写真撮影に由来するアイデアです。ここでも考え方は同じです。ただし、ランダム値を使ってシステムに作用し、固有の結果を 4 つ取得して比較しようというわけです。

後ほど、ランダムな Wedge 値が記載されたオプションすべてを含む、モザイクをレンダリングします。



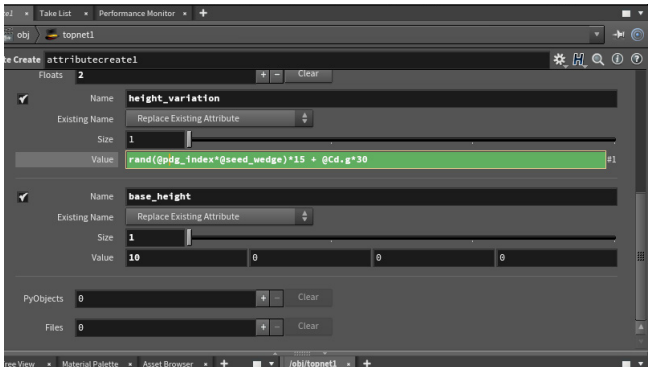
04

HDA Processor ノードを選択し、**HDA Parameters** タブで **Center** パラメータを次のように設定します。

```
@center_wedge.0, @center_wedge.1, @center_wedge.2
```

make_citygrid HDA Processor ノードを選択して **Shift + V** を押し、ノードを ***Dirty (変更あり) にしてクック***します。すると 4 つの都市グリッドが作成されます。

street_grid オブジェクトを非表示にします。ワークアイテムをクリックし、それぞれのグリッドを視覚化します。グリッドごとに、都心部が配置されている緑色の位置が異なります。



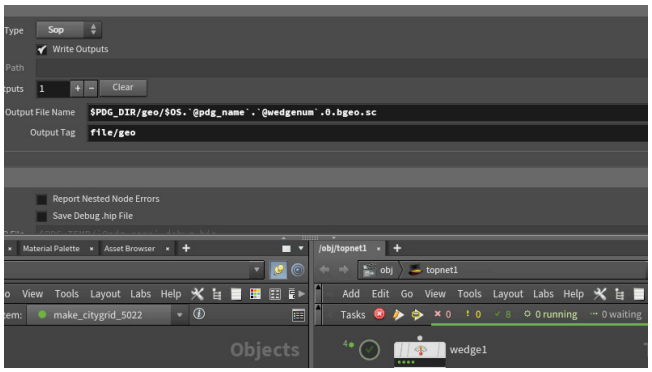
05

高さのバリエーションを Wedge 化して、異なる 4 つのルックが得られるようにします。***wedge*** TOP を選択し、**Wedge Attributes** の横にある **+ (プラス) 記号**をクリックして、2 番目のアトリビュートの **Attribute Name** を ***seed_wedge*** に設定します。**Type** は **Float** のままにして、**Start/End** を **0, 1000** に設定します。

Random Samples パラメータを**オン**にします。

最初の attributecreate TOP で、***height_variation*** アトリビュートの **Value** を次のように変更します。

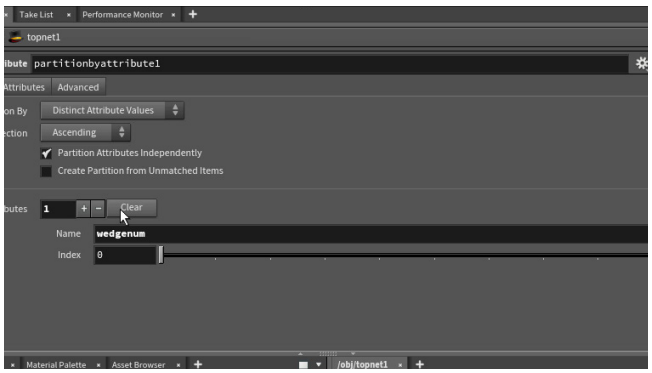
```
rand(@pdg_index*@seed_wedge) * 15 + @Cd.g*30
```



06

出力ファイルを作成するすべての TOP について、出力ファイル名を更新し、Wedge 番号が含まれるようにする必要があります。そうすれば、Wedge 番号でファイルを区別できるようになります。次の TOP ノードの出力ファイルパラメータで、`.bgeo` の直前に `.`@wedgenum`` を挿入します。

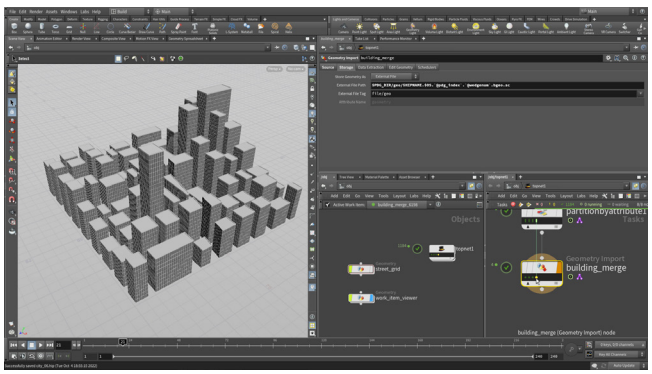
- ***make_citygrid*** HDA Processor
- ***geometryimport*** Geometry Import
- ***create_buildings*** ROP Geometry
- ***building_merge*** Geometry Import



07

Wait for All ノードを **Partition by Attribute** TOP ノードに置き換えます。**Partition By** が **Distinct Attribute Values** に設定されていることを確認してください。

Attributes の横にある **+ (プラス) 記号**をクリックし、**Name** を ***wedgenum*** に設定します。これにより、Wedge ごとにパーティションが作成されます。



08

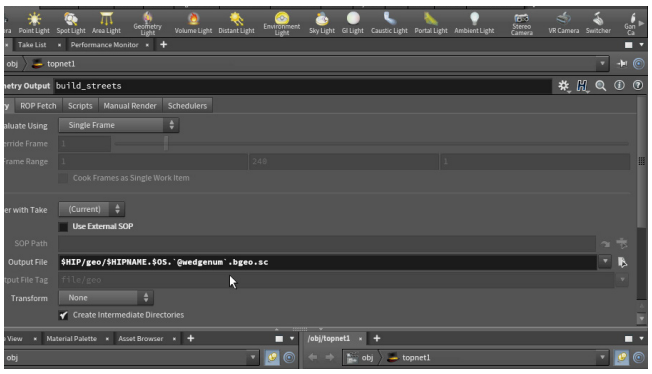
作業内容を**保存**してから、***building_merge*** TOP ノードをクックします。都市マップが 4 つ作成され、***building_merge*** ノードのワークアイテムをクリックすることで視覚化できます。

ここでは Local Scheduler が使用されているので、処理に少し時間がかかります。スケジューラを使用してタスクを演算ファームに分散すると、処理速度がかなり上がります。

パート 8

街路のジオメトリの作成

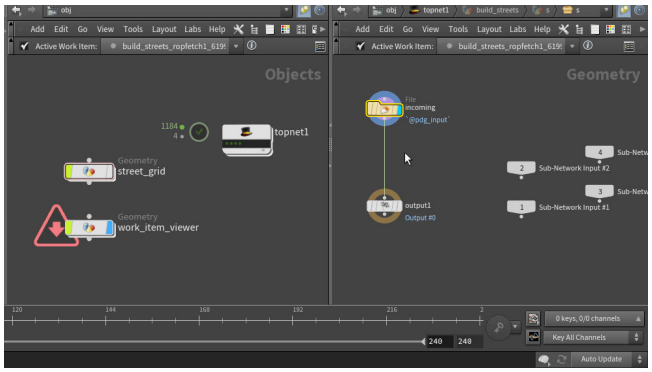
ビルのコンテキストを作成するため、最終レンダリングで使用する都市ブロックと街路を作成します。4 つの Wedge には、現時点ではすべて同じマップが使用されています。このジオメトリを TOP でセットアップして、後で異なるマップを使用できるようにします。非常に堅牢なシステムを作るには、柔軟性を組み込むことが大切です。



01 TOP ネットワークで、**ROP Geometry Output** ノードを追加し、***make_citygrid*** HDA Processor から分岐させます。新しいノードの名前を ***build_streets*** に変更します。

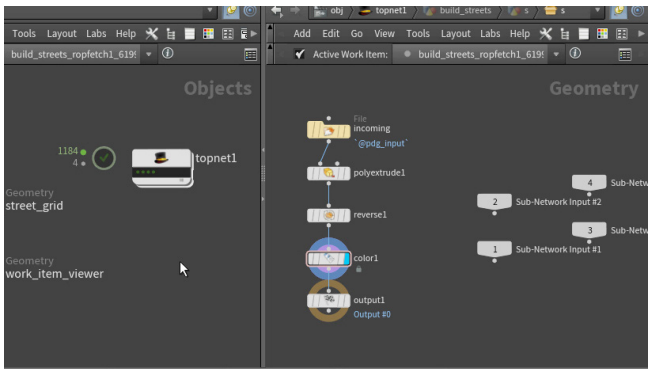
Use External SOP オプションを**オフ**にし、**Output File** のエクスプレッションで、.bgeo の直前に $F ではなく `@wedgenum` を追加します。

ROP Fetch タブで、**Cache Mode** を **Write Files** に設定します。



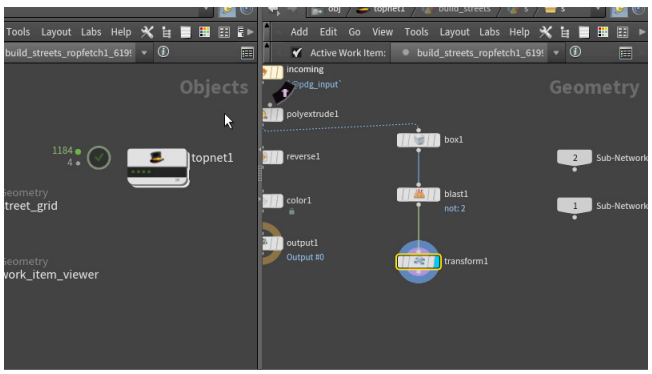
02 このノードを **RMB クリック**して **Generate Node** を選択し、4 つのワークアイテムを作成します。いずれかをクリックしてワークアイテムをハイライトしたら、ノードをダブルクリックしてその中に入ります。

するとジオメトリレベルに移るので、そこで街路を作成していきます。



03 ***incoming*** ノードと ***output*** ノードの間に **PolyExtrude** ノードを追加します。**Distance** を **-0.05** に設定します。**Output Front** チェックボックスを**オフ**にし、**Output Back** オプションを**オン**にします。

その後に **Reverse** ノードを追加して法線を修正したら、**Color** ノードを追加してすべての都市ブロックを白色にします。

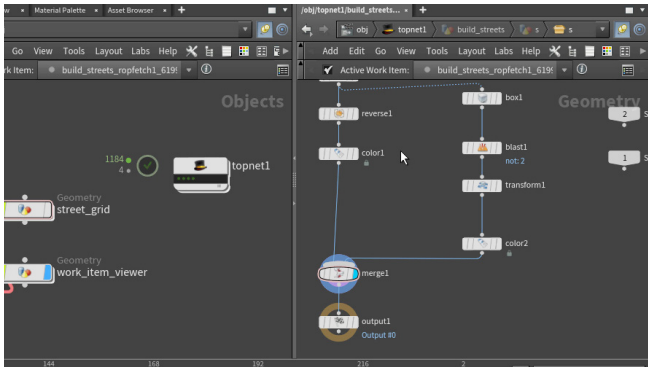


04 ネットワークの端の方に **Box** ノードを追加します。box ノードに ***polyextrude*** を接続し、境界ボックスを都市グリッドのジオメトリに合わせます。

box の後に **Blast** ノードを追加し、**Group** を **2** に設定して、**Delete Non Selected** オプションを**オン**にします。

blast の後に **Transform** ノードを追加し、次のように設定します。

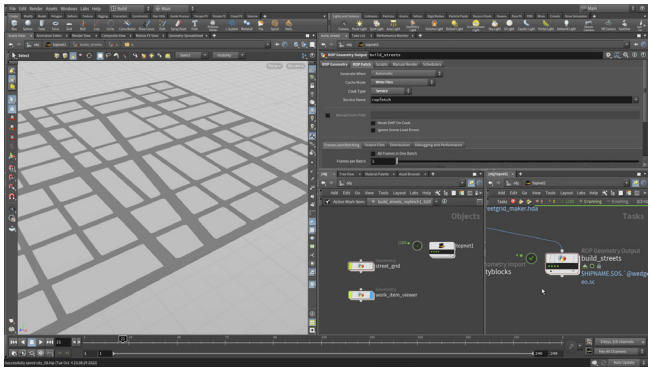
- **Translate Y** を **-0.05** にする
- **Scale X** を **1.05** にする
- **Scale Z** を **1.05** にする



05

Color ノードを追加し、ミディアムグレー (0.33, 0.33, 0.33) に設定します。

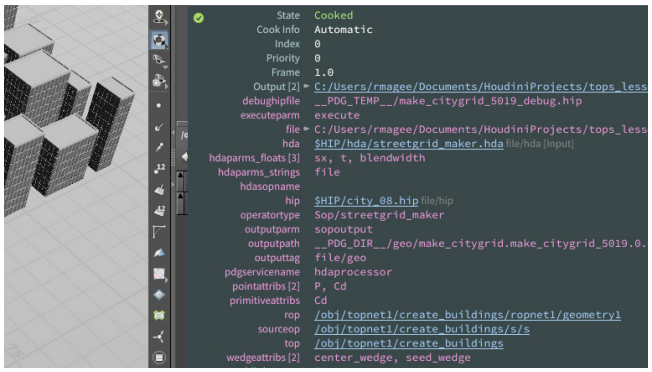
Merge ノードを作成し、2 つの ***color*** ノードに接続します。街路のジオメトリが表示されるよう、**Display フラグ**が出力に設定されていることを確認します。



06

シーンファイルを**保存**します。TOP ネットワークに戻り、ノードをクックします。ワークアイテムをクリックすると、グリッドが表示されます。

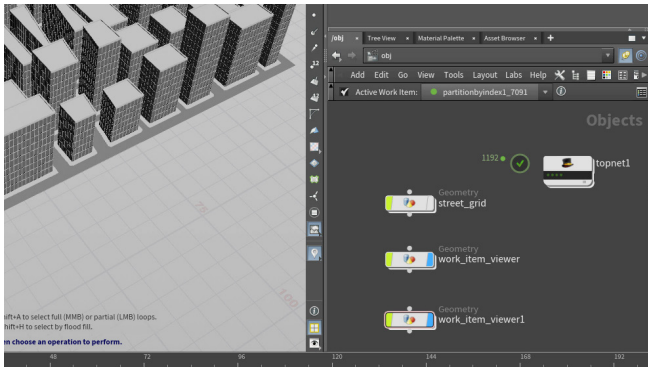
現時点では、これら 4 つはすべて同じです。方程式に都市グリッドを追加すれば、変わっていくはずです。



07

チェーンの終端に **Partition by Index** を追加します。その 1 つ目の入力に ***building_merge*** ノードを接続し、2 つ目の入力に ***build_streets*** を接続します。

新しいノードを**クック**します。完了したら、ワークアイテムをクリックし、ビルのみが表示されることを確認します。ワークアイテムを**中クリック**すると、Partition によって 2 つの出力ファイルが作成されていることを確認できます。新しい出力ファイルを表示して、ビューポートで確認します。



08

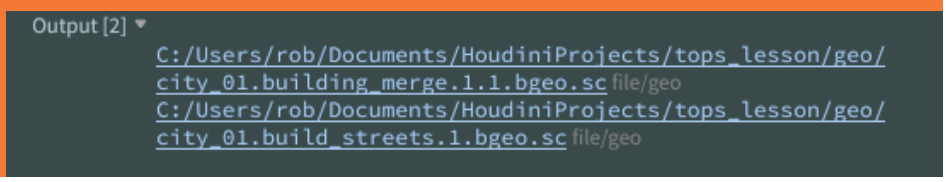
オブジェクトレベルに移動します。***work_item_viewer*** を **Alt ドラッグ**し、***work_item_viewer1*** という 2 つ目のノードを作成します。ジオメトリレベルに入り、Geometry File を次のように設定します。

```
`@pdg_output.1`
```

オブジェクトレベルに上がり、TOP ネットワークで ***partitionbyindex*** ノードのワークアイテムの 1 つを選択します。今度は、両方の出力がビューポートに表示されているはずです。これは、次の手順で都市の画像をレンダリングする際に重要となります。

もう 1 つの PDG 出力

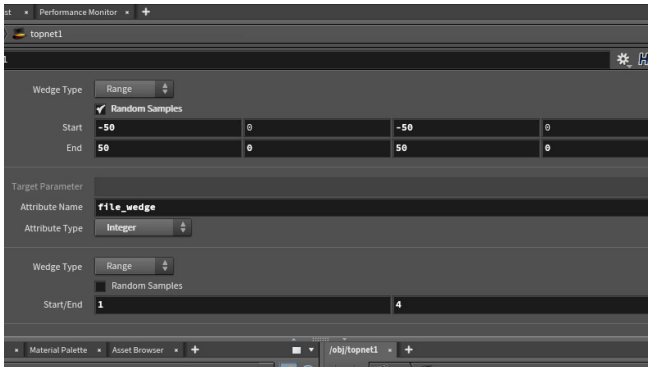
以前は、**pdg_output** を使用して、各ノードの出力結果を視覚化するのに役立つ **work_item_viewer** ノードを作成していました。しかし、今はこのパーティションができたので、実際には 2 つの出力があり、ワークアイテム上を **Ctrl + MMB クリック**して確認することができます。そのため、この 2 つ目の出力が正しく表示されるようにするには、独自のビューアが必要です。



パート 9

4 つの都市マップの Wedge 化

Wedging にもう 1 つ変数を追加してみましょう。4 つの異なるマップにアクセスして、それぞれのマップに対して 1 つのレンダリングを作成します。異なる白黒画像がそれぞれトレースされ、システムへの入力情報として使用されます。これは、パイプラインにコンテンツを追加するもう 1 つの方法です。

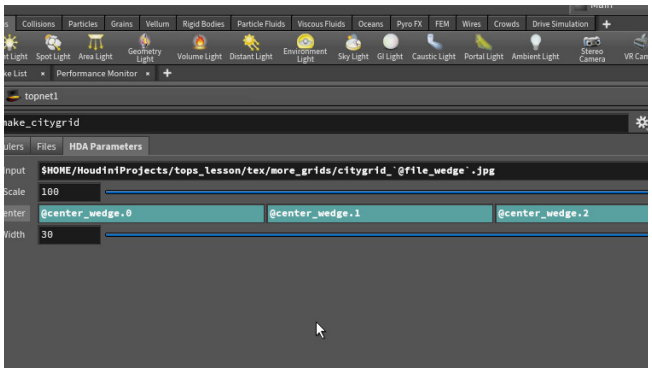


01 ***wedge*** TOP ノードに戻ります。**Wedge Attributes** の横にある **+(プラス)記号**をクリックし、新しい Wedge パラメータを追加します。

次のように設定します。

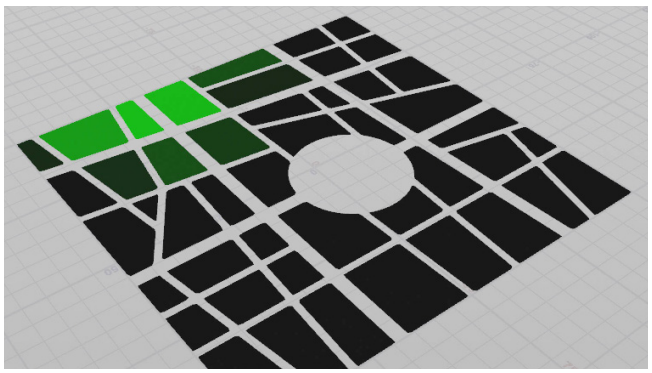
- **Attribute Name** を ***file_wedge*** にする
- **Attribute Type** を **Integer** にする
- **Start/End** を **1, 4** にする

これにより 1、2、3、4 の整数値が作成され、これらを使って異なる都市マップを検索できるようになります。

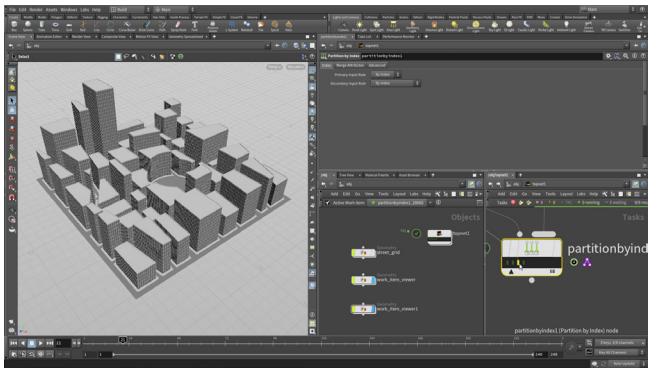


02 ***streetgrid_maker*** HDA Processor ノードで、**Image Input** の横にある**ファイル選択**ボタンをクリックし、***more_grids*** フォルダに移動します。そこからファイルを選択したら、パラメータエディタで、エクスプレッションの `$F` を `` `@file_wedge` `` に変更します。

このフォルダには 4 つのファイルがあり、ネットワークをクックすると、その Wedge アトリビュートが順番にそれらを処理します。



03 ***streetgrid_maker*** HDA Processor ノードを Dirty(変更あり)にしてクックします(**Shift + V**)。4 つの異なるマップが使用され、以前のように都心部があちこち移動するのが確認できます。

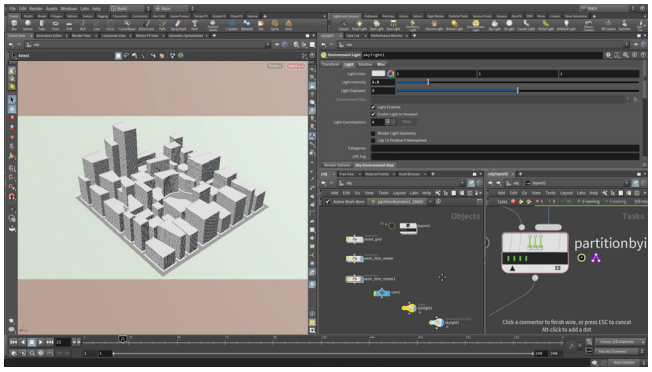


04 2 つ目の ***attributecreate*** TOP ノードを**バイパス**します。これは、静的なマップ向けに設計されたもので、4 つの異なるマップでは正しく動作しません。

最終的なパーティションノードをクックし(**Shift + G**)、4 つの都市の平面図が都市に変換されるのを確認します。

パート 10
モザイクのレンダリング

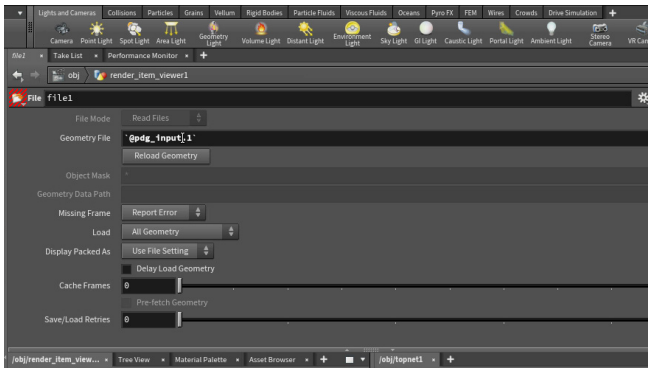
カメラとスカイライトを追加してから、都市をレンダリングします。その結果を単一のモザイクへとまとめますが、Wedge アトリビュートを表示したままにすることで、結果をもとにクリエイティブな決定を下せるようにします。モザイクノードは Image Magick を使用するので、この手順を実行するにはコンピュータにインストールしておく必要があります。



01

都市を斜め上から見下ろすようになるまでビューをタンブルします。**カメラ**メニューから **New Camera** を選択します。4 つの都市グリッドをチェックして、いずれもカメラに収まっていることを確認します。必要に応じてカメラビューを調整します。これを使用してモザイクをレンダリングしていきます。

Skylight を追加します。Environment Light の **Intensity** を **1.3** に設定します。



02

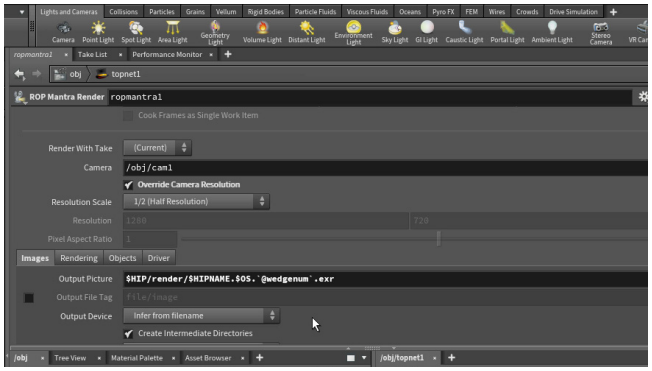
オブジェクトレベルで、***work_item_viewer*** を **Alt ドラッグ**してコピーを作成します。新しいノードを ***render_item_viewer*** という**名前に変更**します。このノードの中に入り、**Geometry File** パラメータを次のように変更します。

```
`@pdg_input`
```

この手順を繰り返して ***render_item_viewer1*** を作成し、その **Geometry File** パラメータを次のように設定します。

```
`@pdg_input.1`
```

これらの ***render_item_viewer*** ノードの Display フラグを設定し、2 つの ***work_item_viewer*** ノードの表示を**オフ**にします。



03

ROP Mantra Render Top を ***partitionbyindex*** の後に追加します。**ROP Fetch** タブで、**Cache Mode** を **Write Files** に設定します。

Camera パラメータが、自分が作成したカメラに設定されていることを確認します。**Override Camera Resolution** オプションを**オン**にして、**½** に設定します **Output Picture** パラメータを次のように変更します。

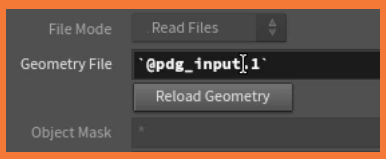
```
$HIP/render/city.$OS.`@wedgenum`.exr
```

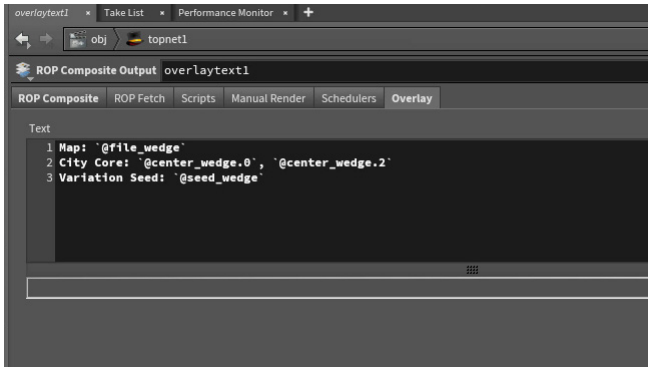
注: Houdini Apprentice を使用してこのチュートリアルを実行している場合は、`.exr` ではなく `.pic` ファイルを使用して、Apprentice のウォーターマークが追加されないようにしてください。

PDG 入力

PDG 出力は、あるノードの結果を取得してそれを表示できるのに対し、PDG 入力は前のノードの結果を取得して、それを代わりに表示します。

この方法で、Mantra ノードは先行するノードの結果をジオメトリの入力として使用した後、そのジオメトリをレンダリングします。これら 2 つのオプションは似ていますが、違いを理解しておくことが重要です。





04

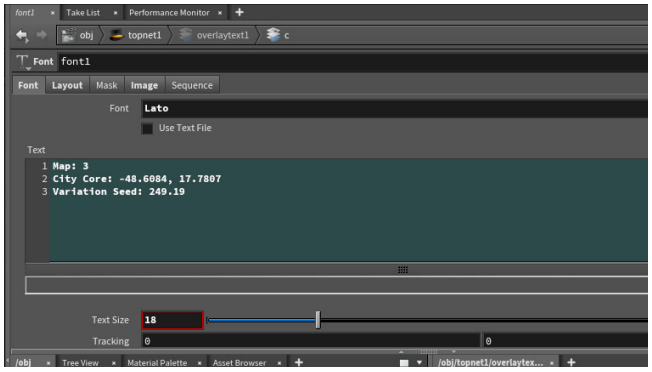
Overlay Text Top を追加し、**Output Picture** を次のように設定します。

```
$HIP/render/$HIPNAME.$OS.`@wedgenum`.exr
```

ROP Fetch タブで、**Cache Mode** を **Write Files** に設定します。

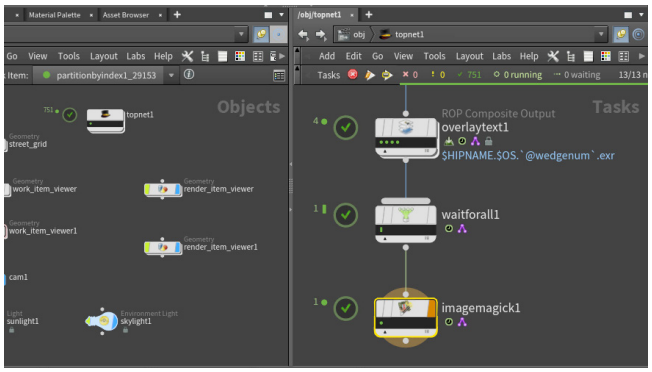
Overlay タブで、次のように入力します。

```
Map: `@file_wedge`
City Core: `@center_wedge.0`, `@center_wedge.2`
Variation Seed: `@seed_wedge`
```



05

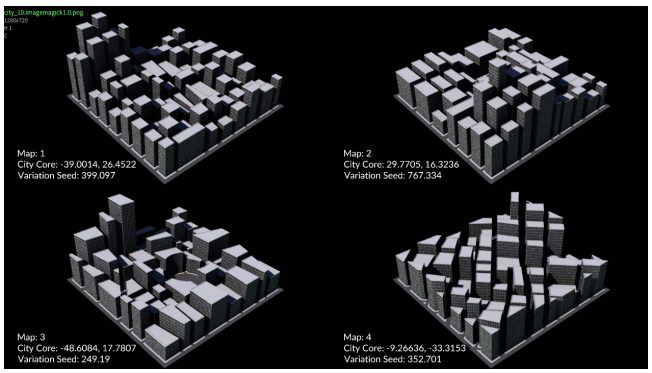
Overlaytext ノードを**ダブルクリック**して、中に入ります。***Font*** ノードを選択し、**Text Size** を **18** に変更します。これでテキストが小さくなり、都市の表示スペースが広くなります。



06

上に戻って、**Wait for All** パーティションノードを追加します。これはすべての要素をまとめるノードです。**ImageMagick** ノードを追加します。

このノードをクックして、TOP ネットワークを処理します。4 つのレンダリングが作成されてから、テキストがオーバーレイされ、1 つのモザイクにまとめられます。



07

完了したら、ワークアイテムを **RMB クリック**し、**View Task Output** を選択します。すると **Mplay** 画像ビューアに最終画像が表示されます。

このコンタクトシートを使用すると、さまざまな Wedge を確認したうえで、どれを採用するかを決定できます。画像上に表示されたパラメータは、希望する都市を得るために確定する必要のある値です。

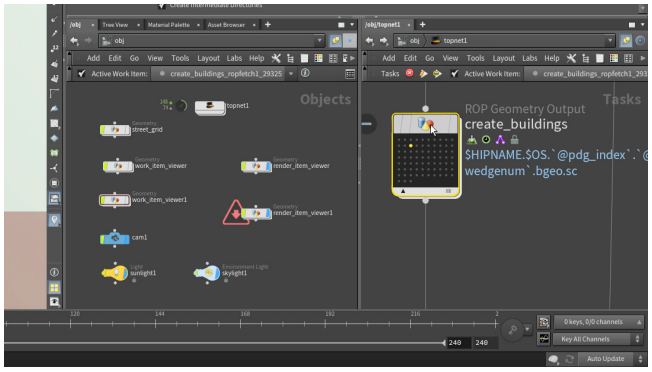
Houdini 以外にも使える PDG

PDG は Python に対応しているため、Houdini と直接接続していないタスクにも使用できます。ワークフローを整理する TOP ノードによって、パイプラインツールを整理するための視覚ツールを提供します。

これらのネットワークは Houdini を通じて処理し、選択したスケジューラに送信できます。作業を演算クラウドに送信すれば、処理を高速化できます。

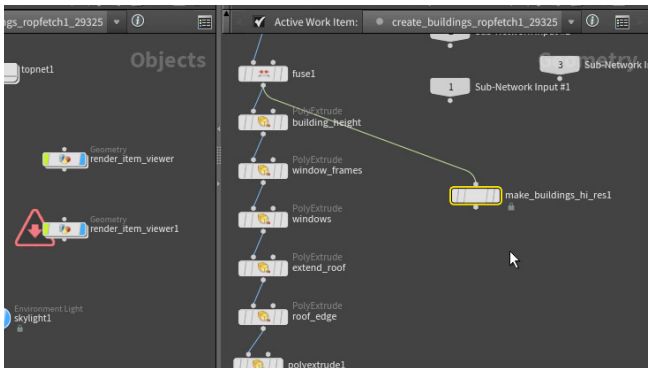
パート 11
スケールアップでより多くのコンテンツを作成

冒頭で、この都市ビルのサンプルは SOP で割と簡単に作成できるだろうと説明しました。しかし問題は、システムが複雑になるにつれてボトルネックが発生することです。これは、すべての処理を単一のネットワーク内で実行することによります。TOP なら、演算ファームにワークアイテムを分散でき、スケールアップしても処理速度は低下しません。



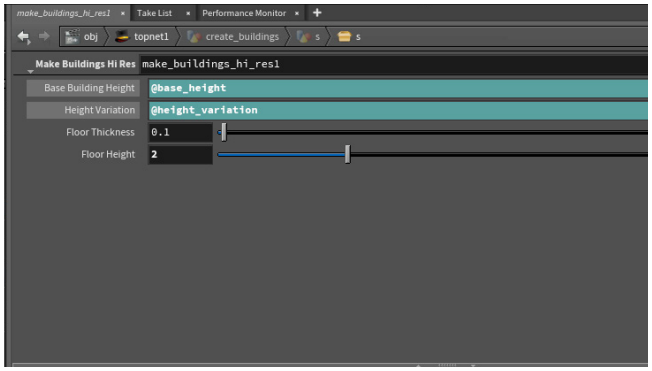
01 Wedge TOP で、Wedge Count を 1 に設定します。これにより、新しいマップを使用して都市をスケールアップする間、反復回数が 1 回のみになります。

create_buildings ROP Geometry TOP ノードを選択し、**RMB クリック**して **Generate Node** を選択します。次は、このノードの中に入り、デジタルアセットを使用して、床、窓、柱を含むさまざまなタイプのビルを作成します。



02 ***ダブルクリック***して中に入ります。**Assets** メニューから **Install Asset Library** を選択します。***hda*** ディレクトリに移動し、***make_buildings2.hda*** を選択します。**Accept** をクリックしてから Install（Install and Create ではありません）をクリックします。**Tab > Make Buildings Hi Res** を選択して、ノードをグラフに配置します。

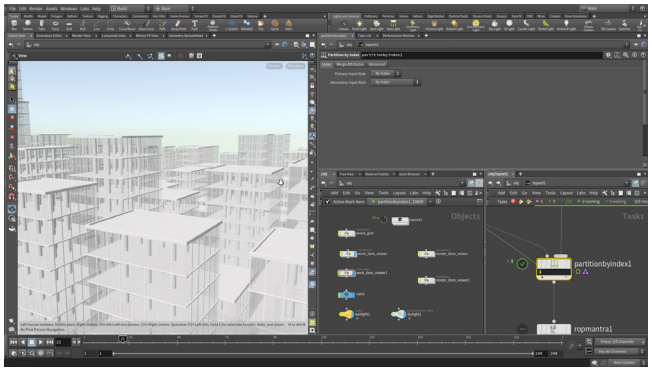
fuse ノードを新しいアセットに接続します。



03 ***make_buildings_hi_res*** デジタルアセットで、次のように設定します。

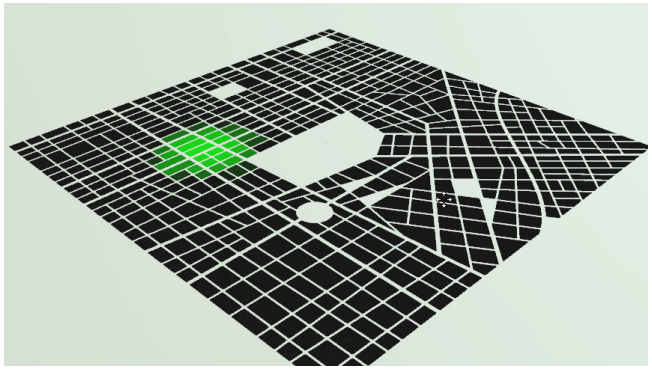
- **Floor Height** を **2** にする
- **Base Building Height** を `@base_height` にする
- **Height Variation** を `@height_variation` にする

color ノードと ***output*** ノードの間に ***switch*** ノードを追加したら、2 つ目の入力に ***make_buildings_hi_res*** ノードを接続し、**Select Input** を **1** に設定します。



04 **Partition by Index** ノードを選択し、ネットワークを**クック**して、新しいビルを表示します。ビルに床板、柱、窓がつきました。

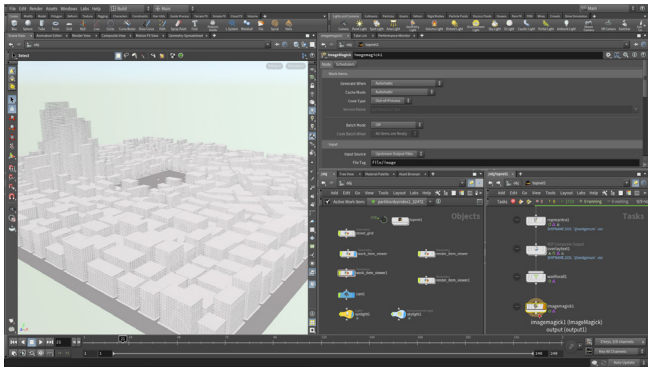
多くのジオメトリが生成されました。演算ファームを使用してワークアイテムの数を増やす利点がわかり始めたのではないでしょうか。他のスケジューラ系ノードの 1 つを使って、それを実現できます。



05

別の都市グリッド画像を使用して、都市を大きくすることもできます。その場合、***streetgrid_maker*** HDA Processor で **Uniform Scale** の値を **500** に上げ、**Image Input** を ***citygrid_large*** マップに変更する必要があります。また、**Blend Width** の値を上げて、都心部の高さの変化を大きくしてもよいでしょう。

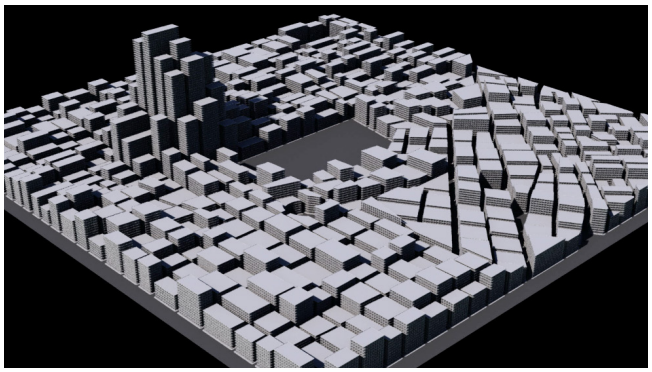
Wedge ノードで、***center_wedge*** アトリビュートを -110, 0, 100 などに設定することもできます。



06

streetgrid_maker HDA Processor ノードを **Dirty（変更あり）にしてクック**します。新しい都市マップが生成される様子を確認できます。完了後は、元のマップよりもずっと多くの都市ブロックが生成され、ビルの数もずっと増えています。

この時点で、演算ファームを利用することが理にかなっているのは明らかです。1 台のコンピュータでは、TOP の並列処理機能を十分に活用できないからです。



07

最後のノードを**再クック**し、都市の画像をレンダリングします。これまで構築したシンプルなシステムが、どんなサイズにも拡張できる可能性を持っていることをお分かりいただけたと思います。

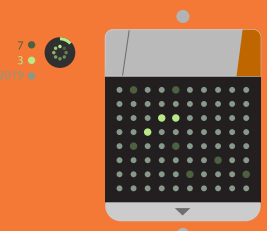
シーンを**保存**します。

まとめ

このレッスンでは、TOP ノードを使って都市マップを取得し、都市ブロックごとにビルを作成した後、このシステムを拡張してより複雑なビルや大規模な都市マップを処理できるようにしました。このプロジェクトでは、TOP ベースのワークフローでよく使用されるノードを紹介し、それらのノードを使ってワークアイテムのタスクを作成および処理する方法について説明しました。

TOP を使って作成可能なものは数多くあり、これはほんの始まりにすぎません。このネットワークタイプを使用すると、一般的な Houdini ワークフローを自動化および処理できるだけでなく、Houdini がまったく関与しないワークフローも処理することができます。

TOP は、効率を求める小規模なスタジオから、膨大なデータを管理する大規模なスタジオまで、パイプライン全体の依存関係を管理できる優れたツールとなるはずです。



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	Q

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
Toggle 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

Lay out 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	Q or B + LMB