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。

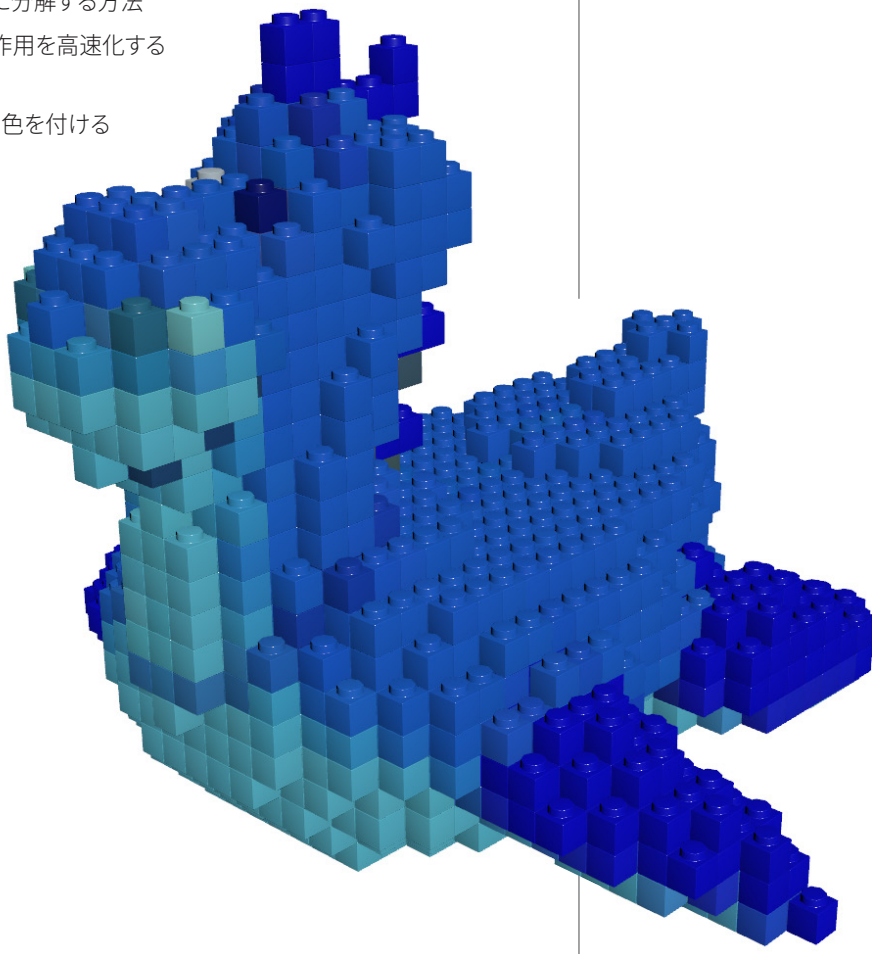
その過程で、Houdini ワークスペースのさまざまな機能を使います。冒頭の概要を参照して、UI 要素の仕組みを確認しておきましょう。このレッスンは、アイデアを実践に移す場です。実践は、最も効果的な学習方法の 1 つです。

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。

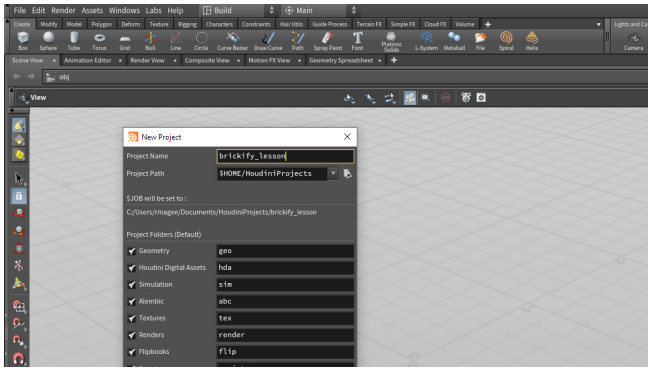
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✔
Houdini Education	✔

ドキュメントバージョン 3.0 | 2022 年 7 月

パート 1

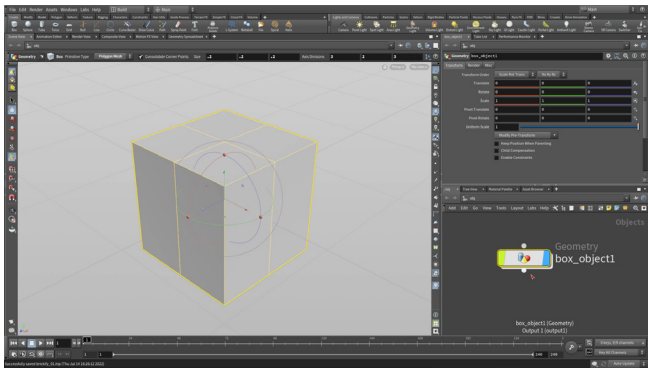
シングルブロックの作成

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。



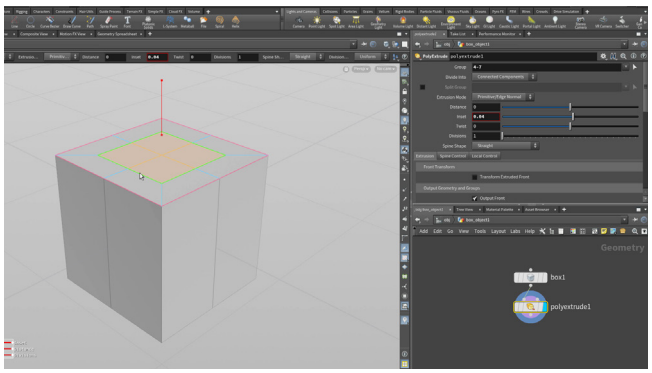
01 **File > New Project** を選択します。**Project Name** を ***brickify_lesson*** に変更し、**Accept** を押します。これにより、プロジェクトディレクトリとサブフォルダが作成され、このショットに関連するすべてのファイルがそこに配置されます。

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。



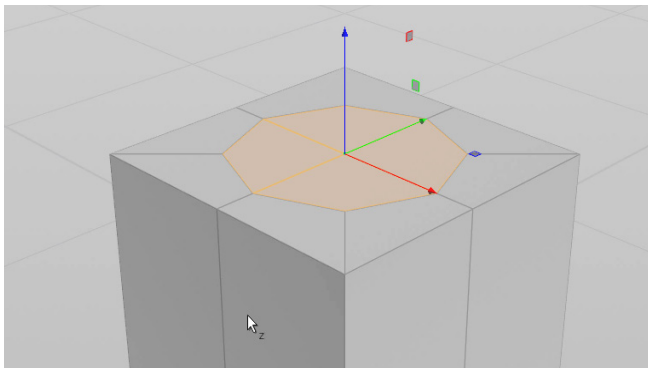
02 ビューポートで、**C を押して** Radial メニューを表示します。メニューから **Create > Geometry > Box** を選択します。カーソルの位置に、シーン内への配置待ちの状態にあるボックスの輪郭が表示されます。**Enter を押して**、原点の位置に配置します。**オペレーションコントロール**ツールバーで、**Size** を **0.2, 0.2, 0.2**、**Axis Divisions** を **3, 2, 3** に設定します。

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。



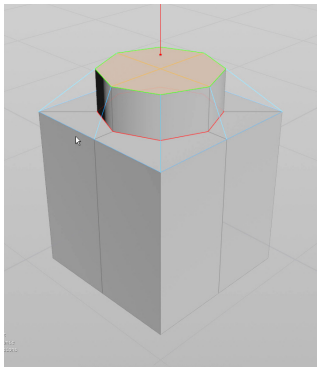
03 **Select** ツールをクリックし、**4 を押して**プリミティブ選択モードに切り替えます。ボックス上部の 4 つのフェースを選択します。**C を押して** Radial メニューを表示し、**Model > Polygons > PolyExtrude** を選択します。ネットワークビューで、***box*** ノードが ***polyextrude*** ノードに接続されているのを確認できます。

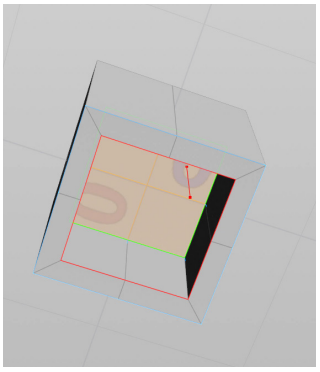
パラメータエディタで、スライダを使用して **Inset** を **0.04** に設定すると、ボックスの上面に新しいポリゴンが作成されます。各ノードには、そのノードの目的に関連するパラメータが含まれています。これらはジオメトリノードで、SOP (Surface Operator) とも呼ばれています。



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。



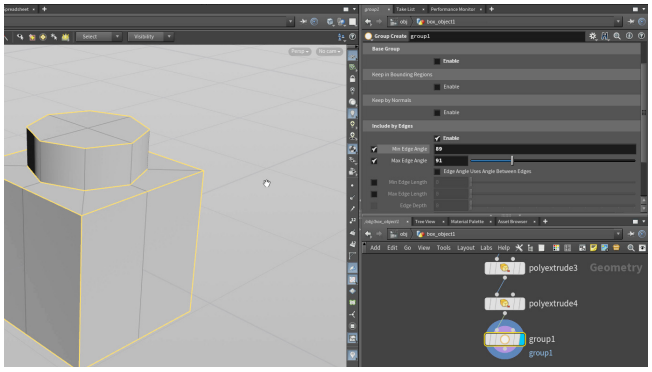


05

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。

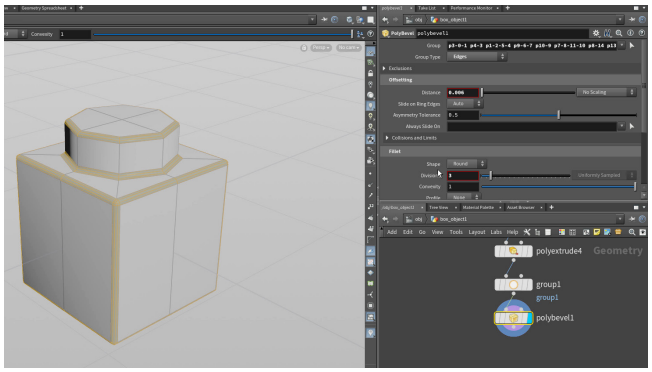
Q を押してツールを繰り返し、**Distance** を **-0.175** に設定します。完了したら、再度タンブルしてブロックの上面が見えるようにします。



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。

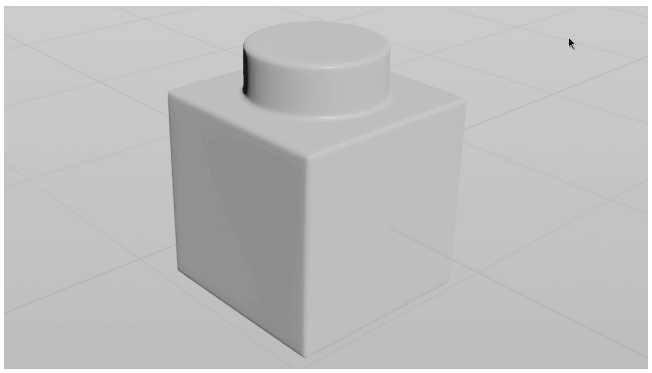


07

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

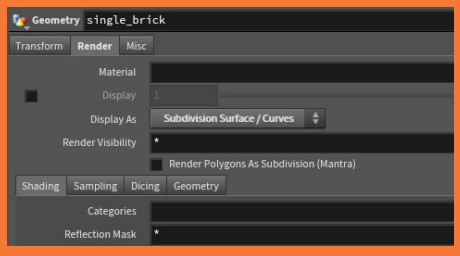
オブジェクトレベルに移動し、ネットワークビューでオブジェクトの名前を ***single_brick*** に変更します。ブロックを選択した状態で、**Shift +** を押して、この形状のサブディビジョンサーフェスの表示をオンにします。ブロックを選択解除して、細分化されたモデルを表示します。オブジェクトにワイヤーラインが見えたら、**V を押し**、Radial メニューから **Shading > Smooth Shading** を選択して非表示にします。

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。

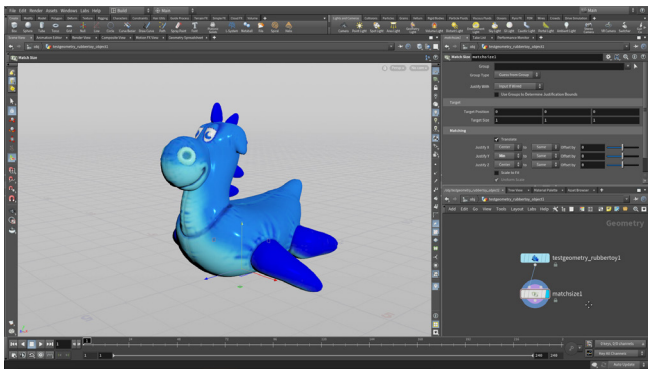
オブジェクトをサブディビジョンとしてレンダリングするには、同じタブで **Render Polygons As Subdivision (Mantra)** をオンにする必要があります。



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。

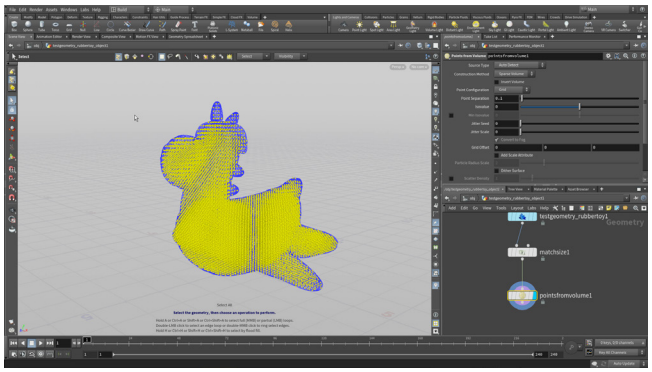


01 Scene View で、Tab を押して **Test...** と入力していきます。**Test Geometry: Rubber Toy** を選択します。**Enter** を押して、原点に配置します。

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。

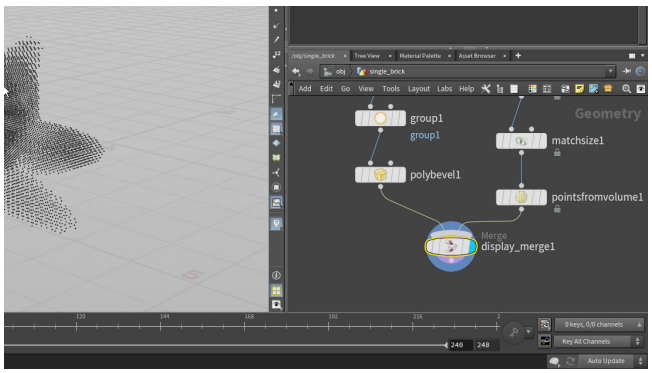
- **Uniform Scale** を **3** にする

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。



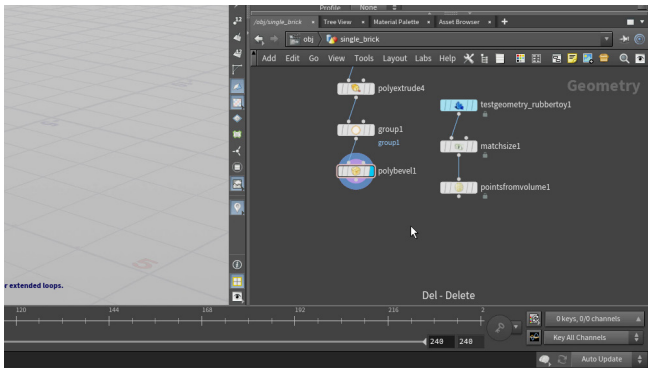
02 ネットワークエディタで、***matchsize*** ノードの出力を **RMB クリック**し、**Points...** と入力して **Points from Volumes** を選択し、そのノードをネットワークに配置します。次に **Display フラグ**を設定して、この新しいノードの出力に集中できるようにします。

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。

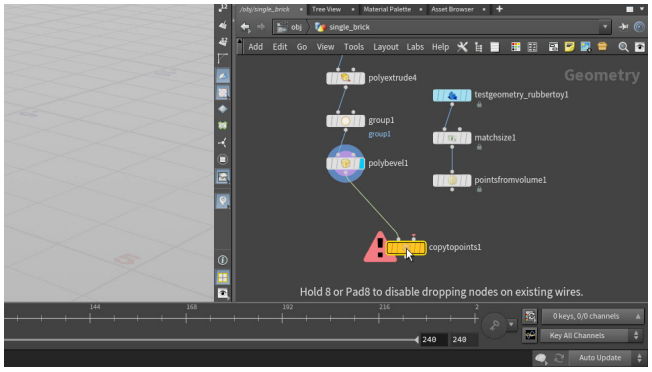


03 **U を押して**オブジェクトレベルに戻り、ゴムのおもちゃの名前を短く ***rubbertoy*** にします。ネットワークビューで **Shift** キーを押し、***rubbertoy*** をクリックしてから ***single_brick*** をクリックします。

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。

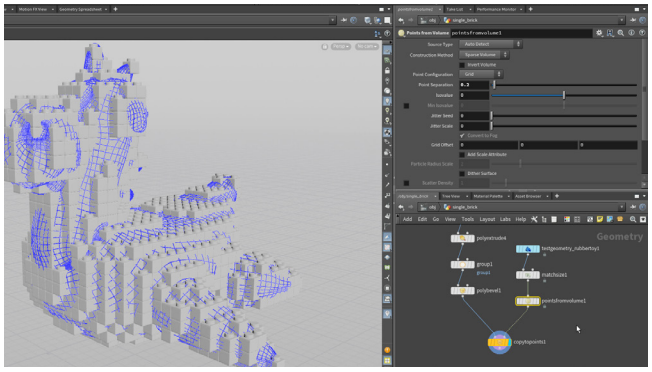


05

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。

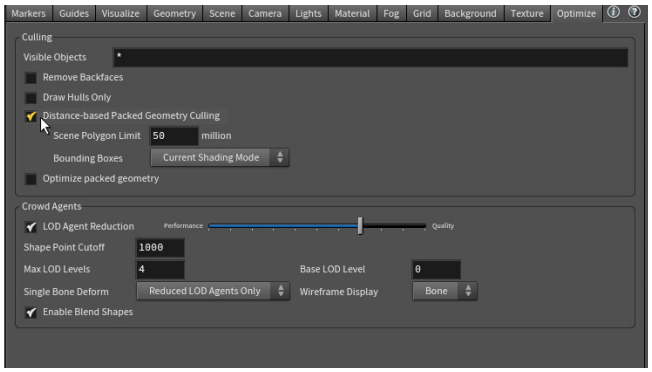


06

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。

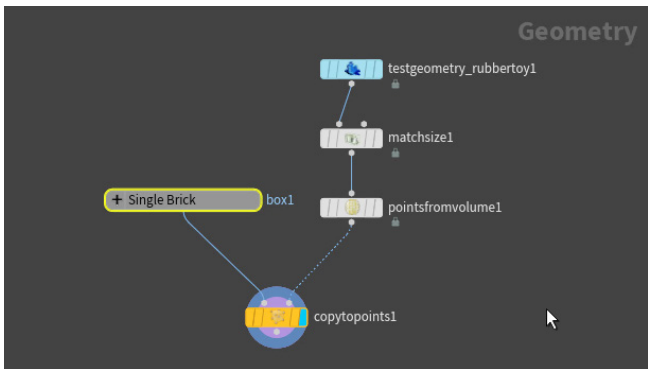
次に、ブロックをグリッドポイントにコピーします。



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

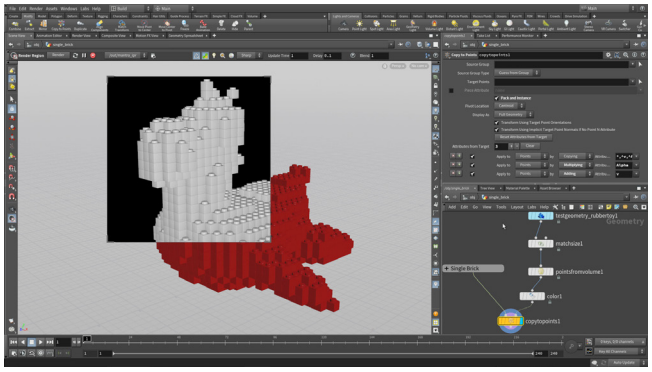
Points	3,032	Center	0, 2.025, 0
Primitives	3,032	Min	-3.1, -0.5, -2.7
Vertices	3,032	Max	3.1, 4.55, 2.7
Packed Geos	3,032	Size	6.2, 5.05, 5.4

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。

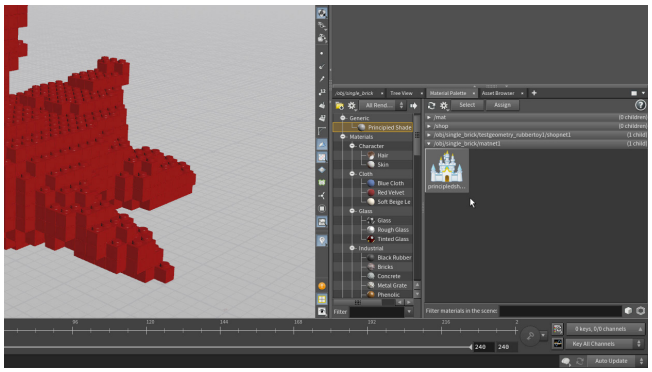


01

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。

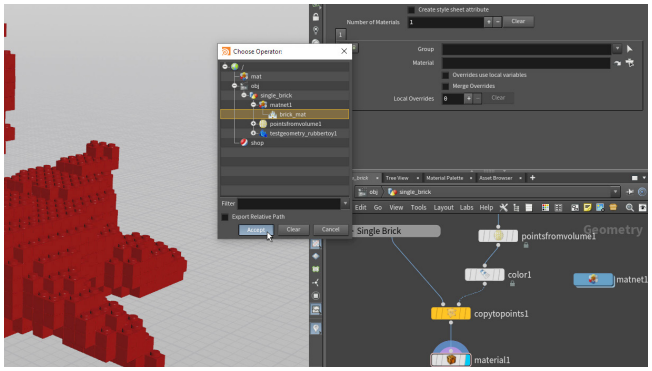
レンダリング領域の右上にある **X** ボタンをクリックして終了します。



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。

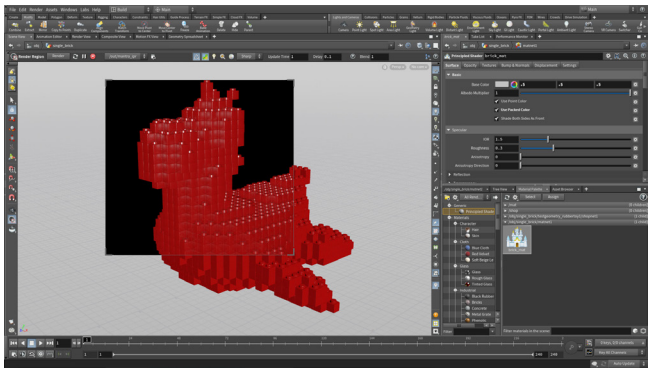


03

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。



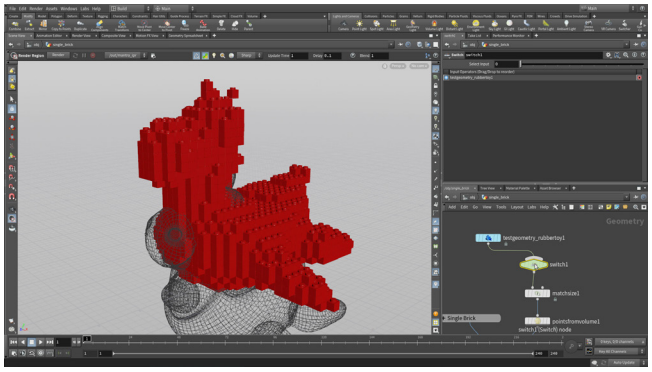
04

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。

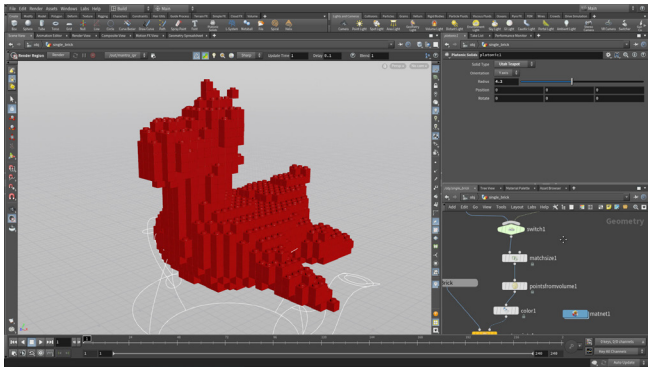
作業内容を**保存**します。レンダリング領域の右上にある **X** ボタンをクリックして終了します。



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。

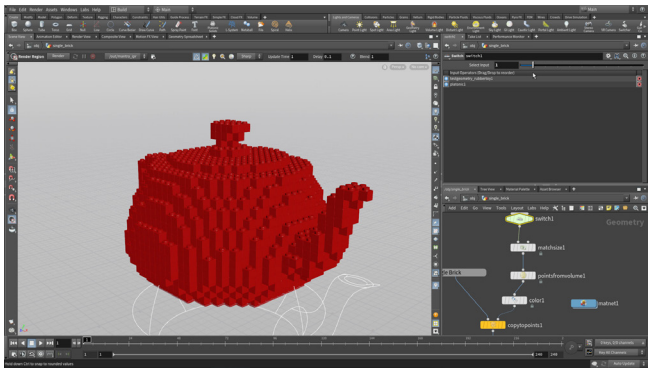
次に、***rubbertoy*** ノードと ***matchsize*** ノードを接続しているワイヤー上にドラッグします。これで、ネットワークの 2 つのノードの間に switch ノードが挿入されます。

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。



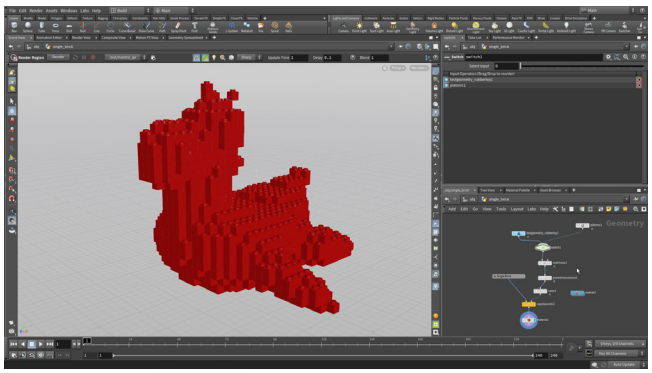
06 ツールシェルフで **Create** メニューに移動し、**Platonic** ツールをネットワークビューにドラッグします。これにより、platonic ノードがジオメトリレベルに配置されます。ツールをネットワークビューにドラッグできるのは、ノードタイプが現在のネットワークレベルに対して有効な場合のみです。

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。



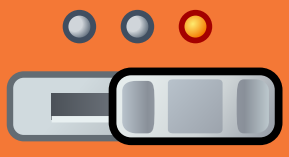
08 ***switch*** ノードで **Select Input** を **0** に設定し、***rubbertoy*** に戻ります。これら 2 つの形状を切り替えたり、さらに形状を追加してブロック化できるようになりました。

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。

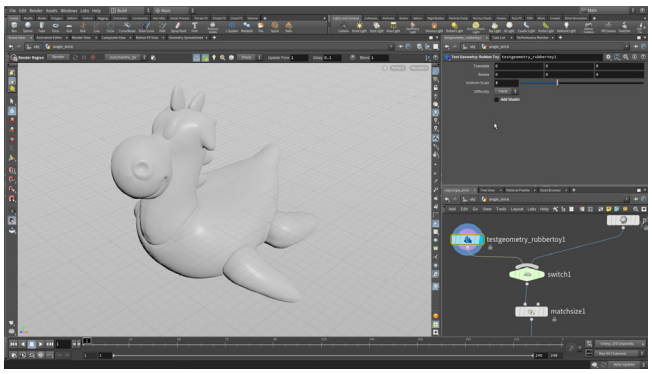
このノードはまた、ネットワークを**デジタルアセット**にまとめる際にも大変便利です。メニューまたはスライダのいずれかとしてアセットにプロモートすれば、さまざまなオプションに素早くアクセスできます。



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

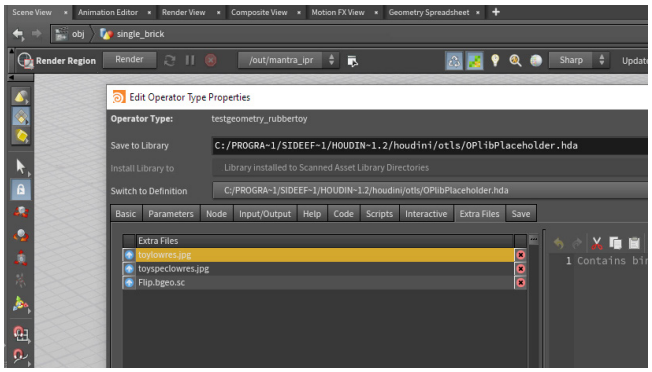
先ほどは、ポイントにカラーアトリビュートを追加して、ブロックインスタンスのカラーに影響するようにしました。
ここでは、単色の代わりにテクスチャマップを使用して、ブロックのルックをさらに面白くしていきます。
いくつか特殊なノードを使用して、テクスチャをポイントカラーに変換します。



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。

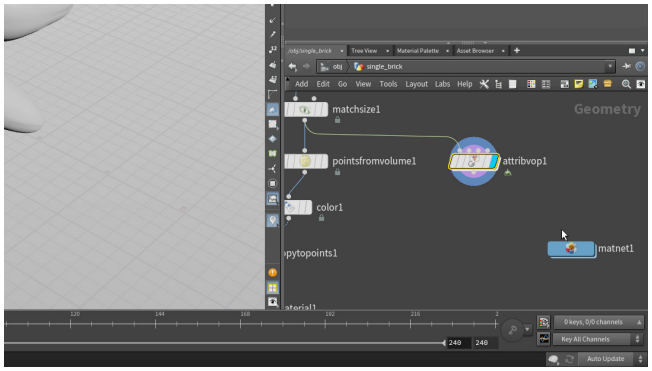
パースビューで UV を非表示にするため、**Display Options** バーに移動して **Show UV Texture when UV's Present** ボタンを**オフ**にします。

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。

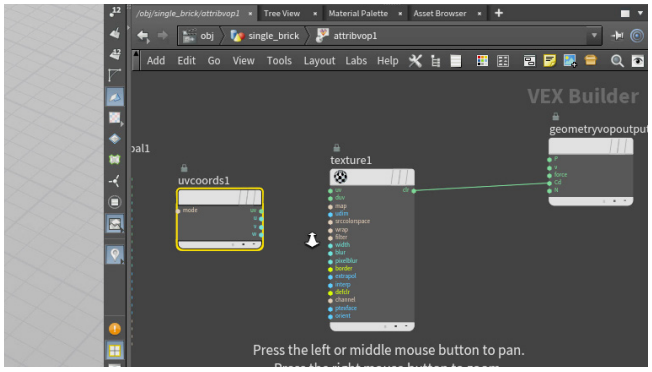
Save as File ボタンをクリックして、***tex*** フォルダに保存します。テクスチャがデジタルアセットに保存されたので、アセットと一緒に共有することができます。ディスク上のテクスチャを使用して、ブロックにカラーを追加していきます。



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。

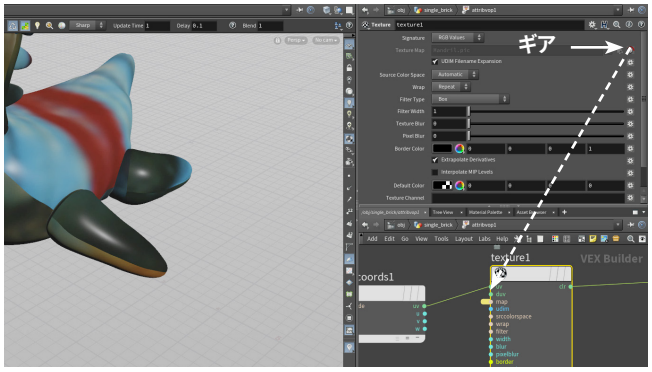
matchsize ノードの出力を、***attributevop*** ノードの 1 つ目の入力に接続します。この新しいノードに **Display フラグ**を設定します。

パラメータエディタで、**Run Over** を **vertices** に設定します。



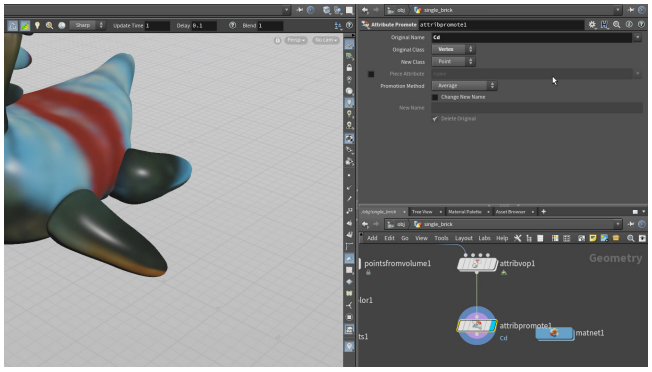
04 ***attributevop*** ノードを**ダブルクリック**して中に入り、**Tab** キーを使用して **Texture VOP** を追加します。それを ***geometryvopoutput*** ノードの **Cd** 入力に接続します。

UV coordinate ノードを追加して、***texture*** ノードの **UV** 入力に接続します。



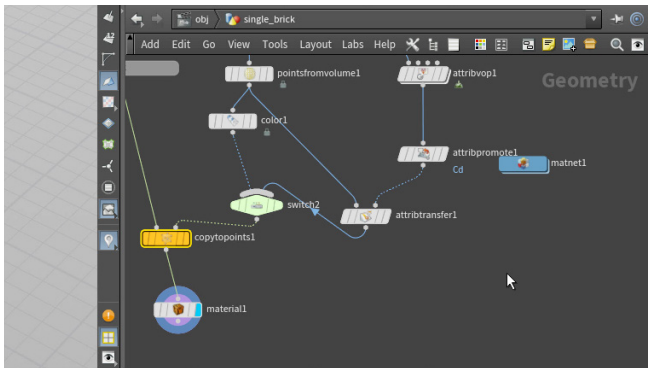
05 ***texture*** ノードを選択します。**Texture Map** パラメータの右端の**ギア** アイコンをクリックして、メニューから **Promote Parameter** を選択します。このパラメータがこのノードの上位レベルに追加されます。

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。

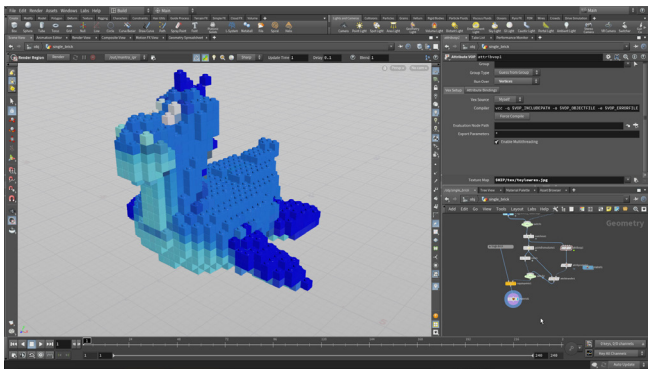
このチェーンの終端に **Attribute Promote** を追加します。**Original Name** を **Cd**、**Original Class** を **Vertex** に設定します。**New Class** の設定は **Point** のままにしておきます。



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。

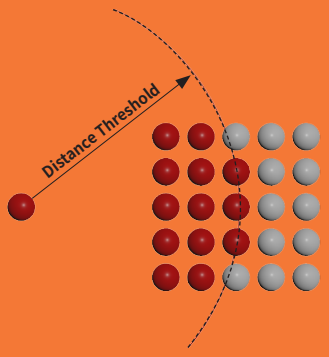
attribvop ノードを選択したら、**Texture Map** パラメータの右端の**ファイル選択**ボタンをクリックして、***toylowres.jpg*** テクスチャに移動し、**Accept** をクリックします。テクスチャマップのカラーが頂点に割り当てられているのを確認できます。

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。

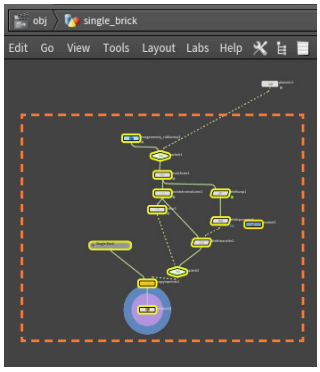
ゴムのおもちゃの場合は、ジオメトリ上のポイントの **Cd** アトリビュートを、ブロックをコピーするのに使用したポイントクラウドに転送しています。UV やキャプチャウェイトなどのアトリビュートもコピーできます。

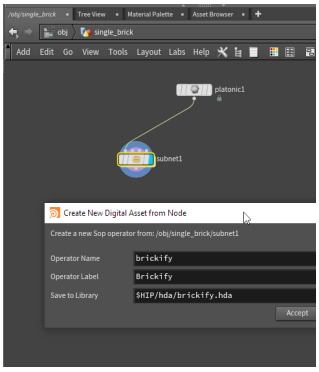


パート 5

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。

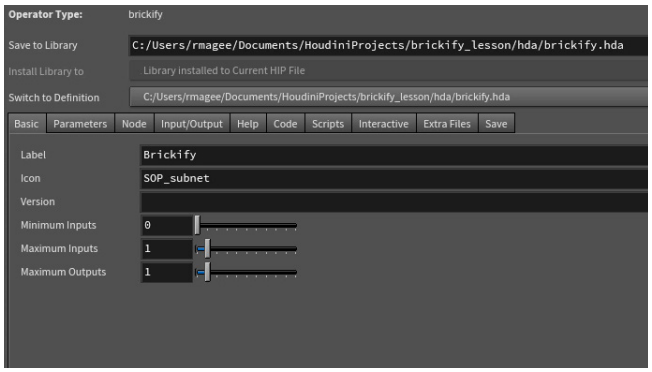




01

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。

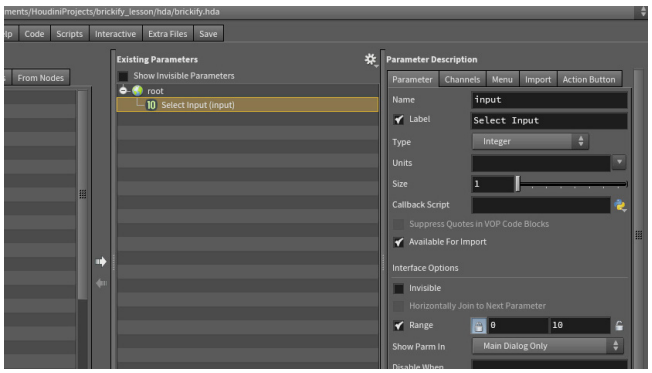


02

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

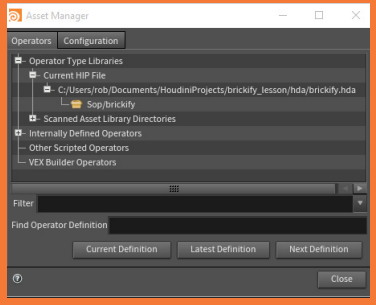
ネットワークビューで、新しいアセットノードを ***brickify_asset*** という**名前に変更**し、**ダブルクリック**して中に入ります。***testgeometry_rubbertoy*** と ***Subnetwork Input*** ノードを切り替えている ***switch*** ノードをクリックします。Subnetwork Input ノードは、上のレベルからプラトン立体のティーポット形状を取り込んでいます。

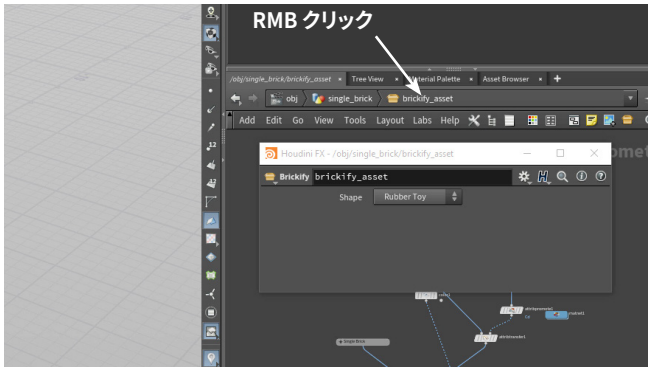
Type Properties ウィンドウで **Parameters** タブをクリックします。**Select Input** パラメータ名をクリックして、それを **Type Properties** ウィンドウの **Existing Parameters** リストに **LMB ドラッグ**します。それを root にドロップして、UI に追加します。**Apply** をクリックするとパラメータが追加されますが、Type Properties ウィンドウは終了しません。

.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。

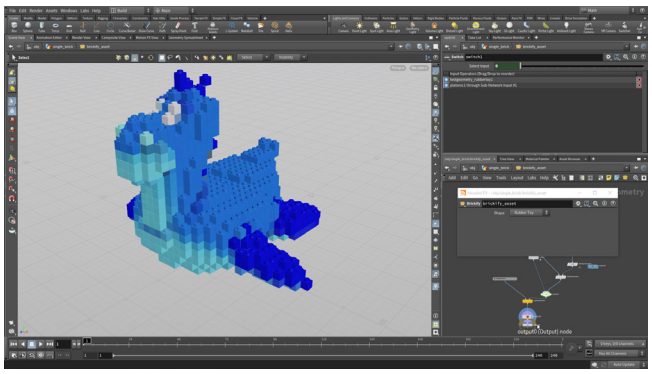
シーンに参照されているアセットを管理するには、**Assets** メニューで **Asset Manager...** を選択し、**Current HIP File** を開きます。





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。

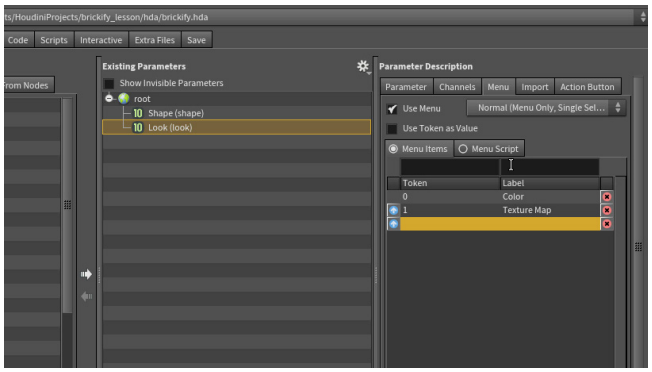
brickify ノードには新しいパラメータがあります。値を **0 から 1** に変更して、シーンへの影響を確認します。問題は、名前があまり適切でないことと、このケースではスライダよりもメニューのほうがずっと効率的に機能することです。そこで、Type Properties を使用して UI を微調整します。



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。

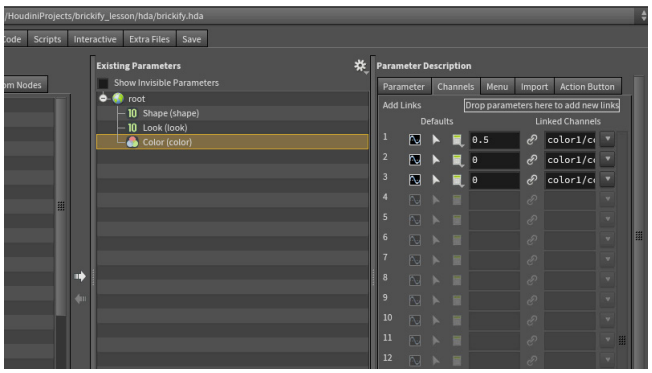
Menu タブをクリックして、**Use Menu** をオンにします。Menu Items で、**Token** に **0**、**Label** に ***Rubber Toy*** と入力して、Enter を押します。次に、**Token** に **1**、**Label** に ***Custom Shape*** と入力します。**Apply** を押します。

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。

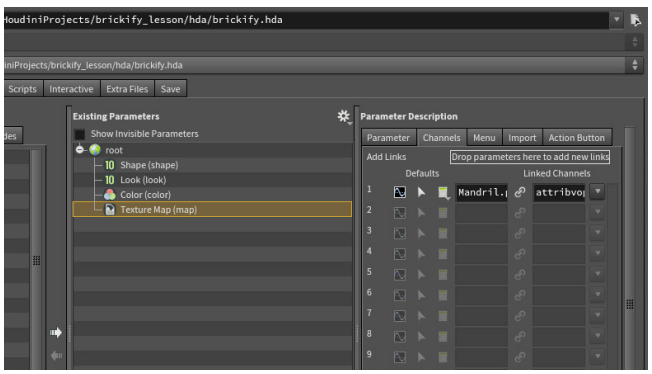
Menu タブをクリックして、**Use Menu** をオンにします。Menu Items で、**Token** に **0**、**Label** に ***Color*** と入力して、Enter を押します。次に、**Token** に **1**、**Label** に ***Texture Map*** と入力します。**Apply** を押します。



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。

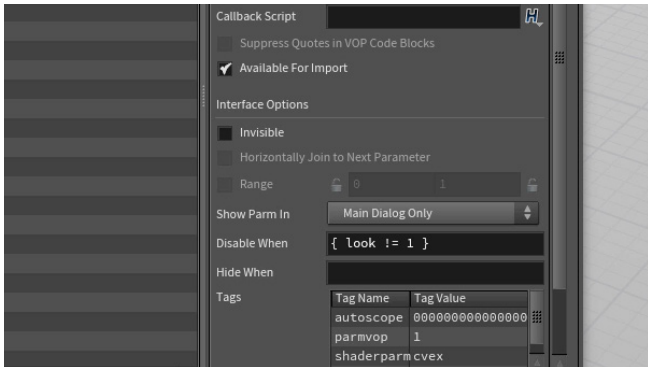
注: 間違って **Accept** を押した場合、新しいパラメータはアセットに保存されますが、**Asset** メニューを使用して **Edit Asset Properties... > brickify** を選択し、再度開く必要があります。



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。

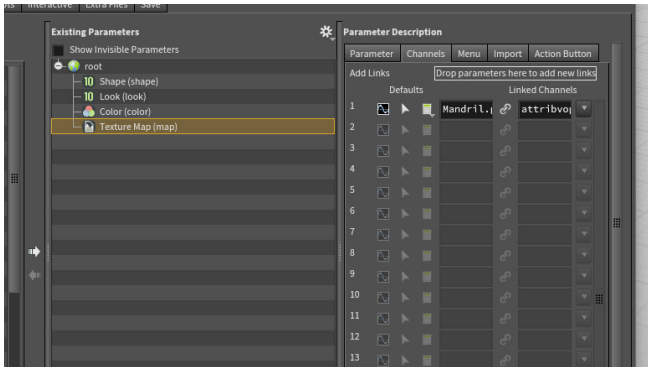
Type Properties ウィンドウで **Apply** を押すと、***brickify_asset*** パラメータインターフェースにこのパラメータが表示され、Mandril テクスチャマップによってブロックに色が付いています。



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。

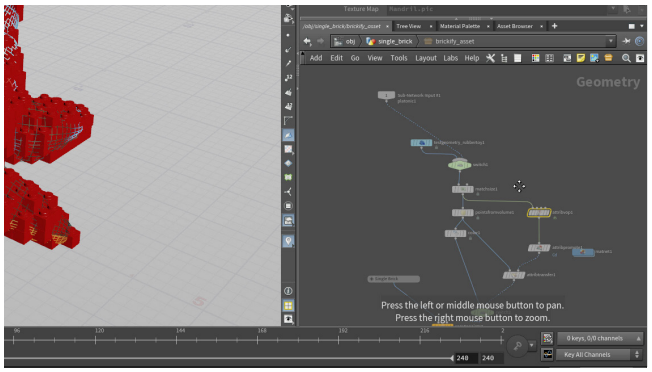
これにより、Look メニューで **Color** が選択されていなければ、このパラメータは無効になります。次に、**Texture Map** パラメータをクリックし、**Disable When** フィールドに `{ look != 1 }` と入力します。

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。

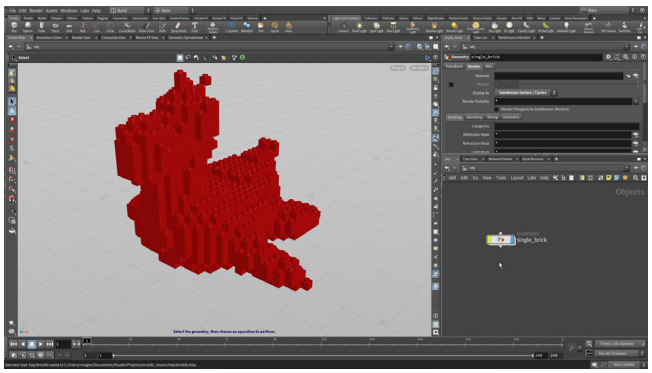
現在、アセットではデフォルトの ***Mandril.pic*** が使用されるようになっています。



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。

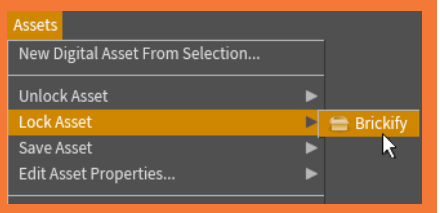
これで、シーンファイルと、シーンがアセットを作成するために参照する .hda ファイルができました。このライブラリを使用すると、このシーンでアセットの他のインスタンスを作成したり、アセットを別のシーンに追加することができます。

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。

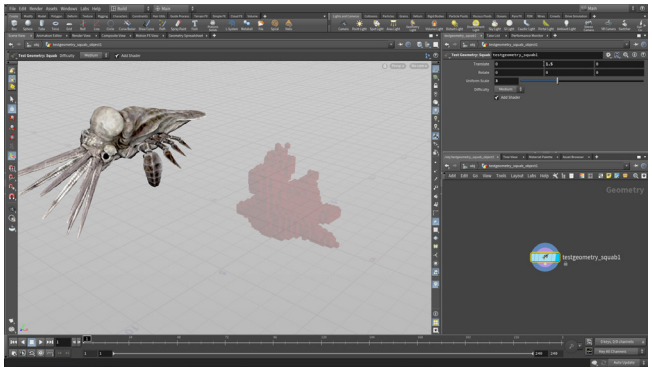
アセットのノードを **RMB クリック**すると、**Allow Editing of Contents** でアセットのロックを解除したり、**Match Current Definition** でアセットをロックできます。ただし、保存するかどうかは尋ねられず、変更が失われる可能性もあるため注意してください。



パート 6

デジタルアセットのテスト

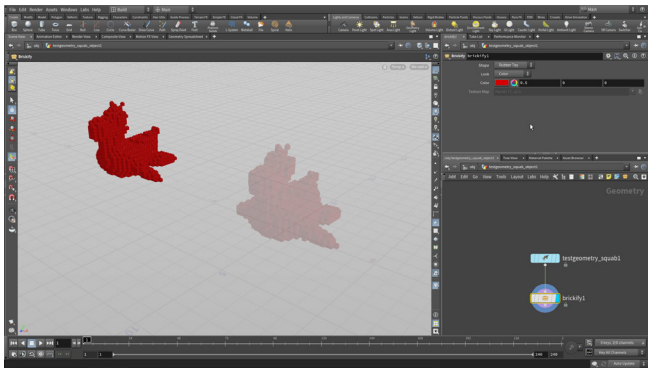
デジタルアセットは、単一のシーンファイルで複数回インスタンス化できます。異なるジオメトリでこのアセットを使用して、動作をテストしていきます。最初のアセットに設定された変更を素早く確認できるよう、テストバージョンを用意しておくことをお勧めします。アセットが適切に動作すれば、他のシーンファイルでも使用できます。



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。

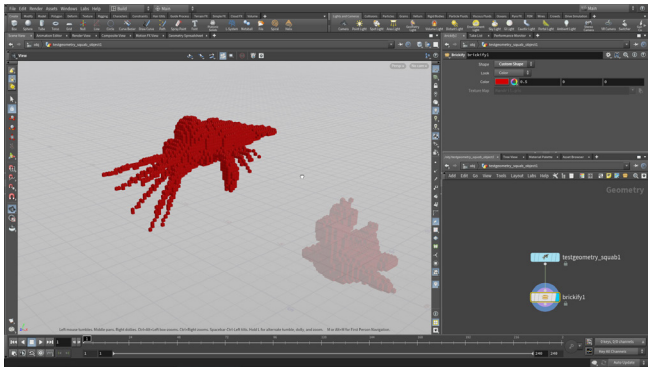
新しいオブジェクトノードを**ダブルクリック**して、ジオメトリレベルに入ります。ノードを**選択**して、**Scale** を **3**、**Translate Y** を **1.5** に設定します。これにより、イカ蟹がゴムのおもちゃよりも少し大きくなります。



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。

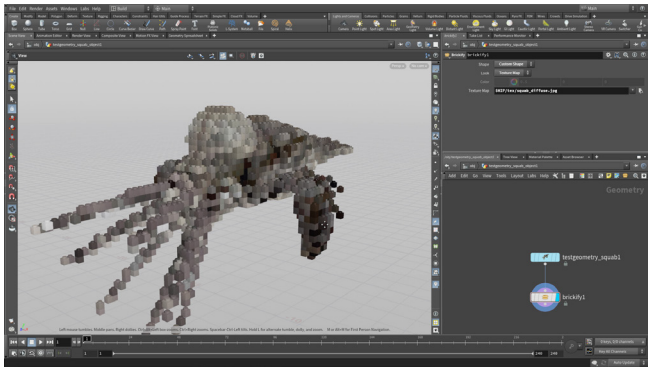
Display フラグを設定すると、**赤色**のゴムのおもちゃがもう 1 つ表示されます。こうなるのは、これらがこのアセットのデフォルトだからです。



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。

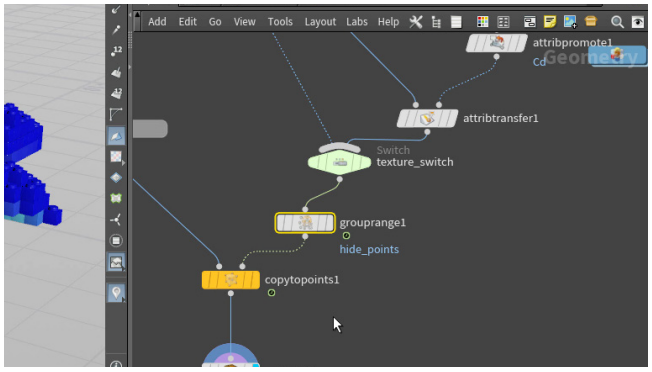
これで、ディスク上のこのテクスチャと、***brickify*** ノードの **Texture Map** パラメータを使用して、ブロックにカラーを追加できるようになりました。

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

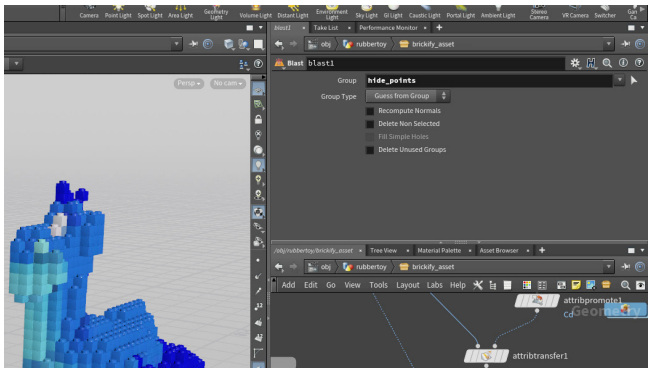
ブロックのアニメート

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。



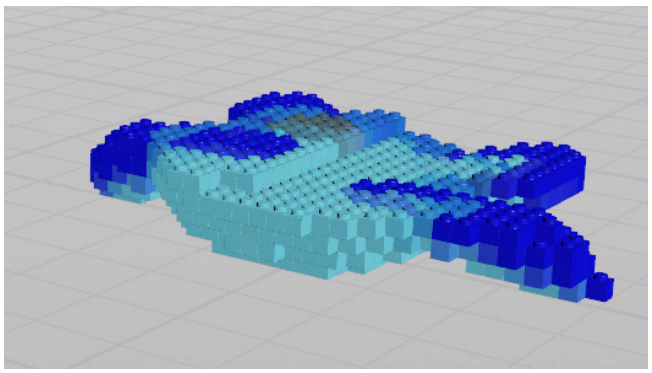
01 ***squab*** オブジェクトを非表示にして、***rubbertoy*** オブジェクトの中に入ります。***brickify_asset*** を選択し、**Assets > Unlock Asset > Brickify** を選択します。***brickify_asset*** ノードを**ダブルクリック**したら、***texture_switch*** の出力を **RMB クリック**して、**Group by Range** を選択します。そのノードを配置して **Display フラグ**を設定したら、次のようにパラメータを設定します。

- **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。

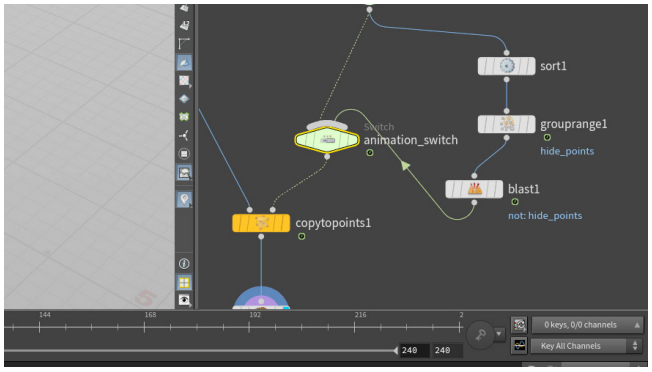
Play を押して、フレーム毎に増えるポイントを確認します。チェーンの終端の ***material*** ノードに **Display フラグ**を設定して、時間とともに増えるブロックを確認します。



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。

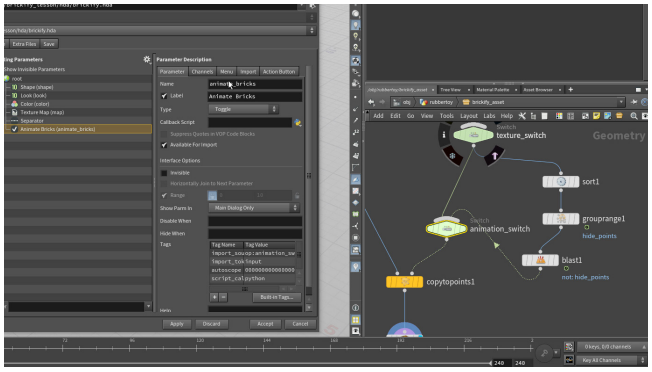
texture_switch ノードの出力を **RMB クリック**し、**Sort...** と入力していき、**Sort ツール**を選択します。このノードを配置したら、**Point Sort** を **Along Vector** に変更します。これを **0, 1, 0** に設定すると、ポイントが下から上に向かって現れるようになります。

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。



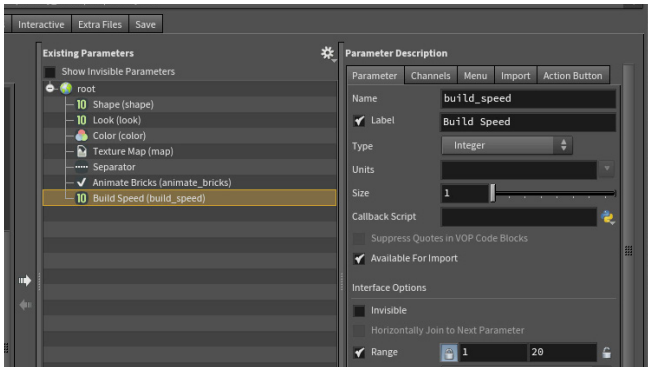
05

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。



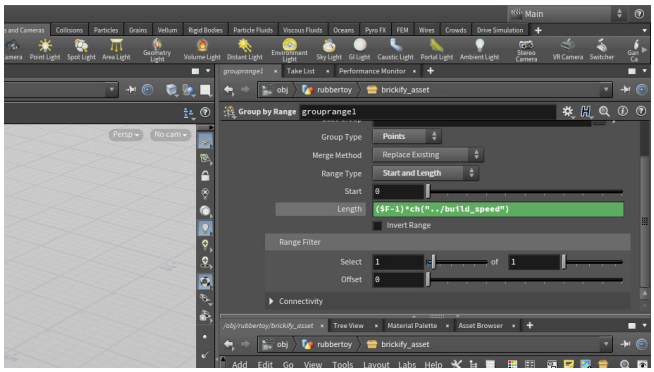
06

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。

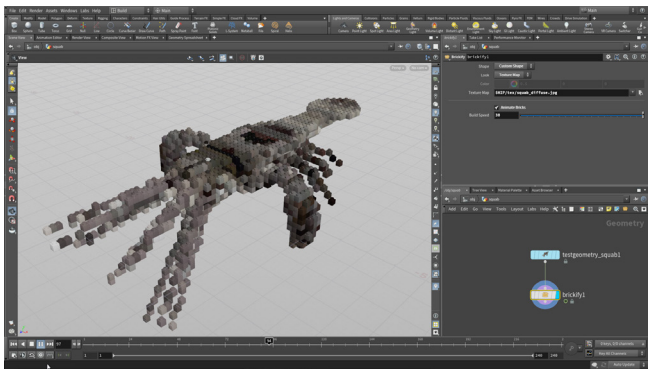


07

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。

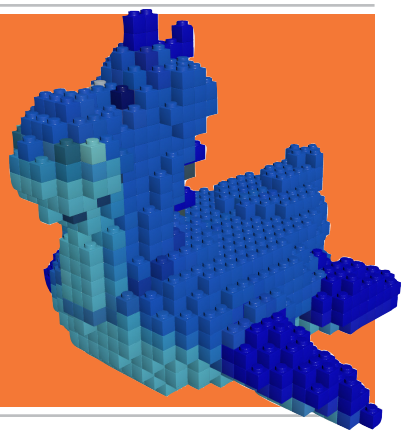
Squab ネットワークに移動すると、***brickify*** で新しい機能が使用できるようになっています。**Animate Bricks** トグルをオンにしたら、この形状はブロックの数が多いので、**Build Speed** を約 **30** に設定します。

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。

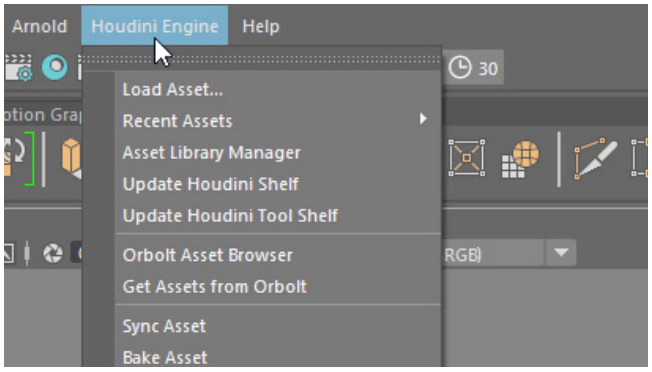
Houdini デジタルアセットは、アーティストがこうしたツールを効果的に共有できるようにするものであり、スタジオレベルのプロダクションを支えます。プロシージャルアセットは反復作業の自動化が簡単にできるため、プロジェクトのクリエイティブなニーズに集中できます。



パート 8

他のアプリケーションに HDA を読み込む

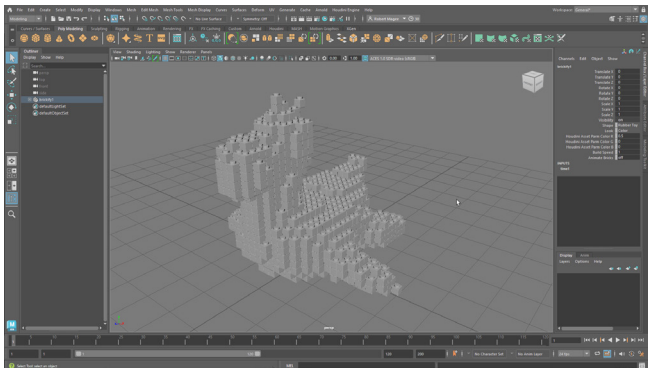
Houdini デジタルアセット（HDA）をディスク上に保存したら、Houdini Engine プラグインを使用して、そのアセットをホストアプリケーションに読み込むことが可能です。こうしたプラグインを使ってアセットを共有すれば、同僚は Autodesk Maya や 3ds Max などの 3D アプリケーションや、Unity や Unreal Engine などのゲームエディタに直接アセットを読み込めるようになります。



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。

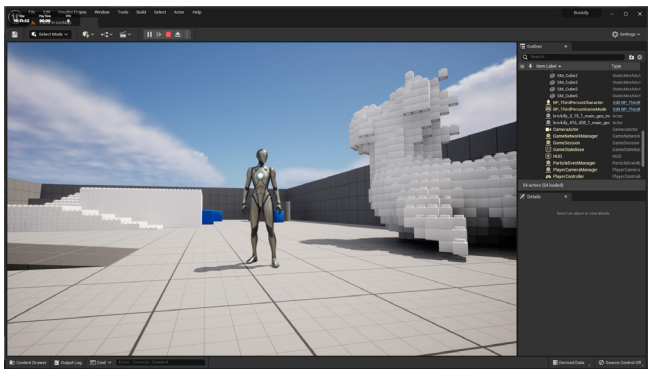
詳細は、**SideFX.com/engine** を参照してください。

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。