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。

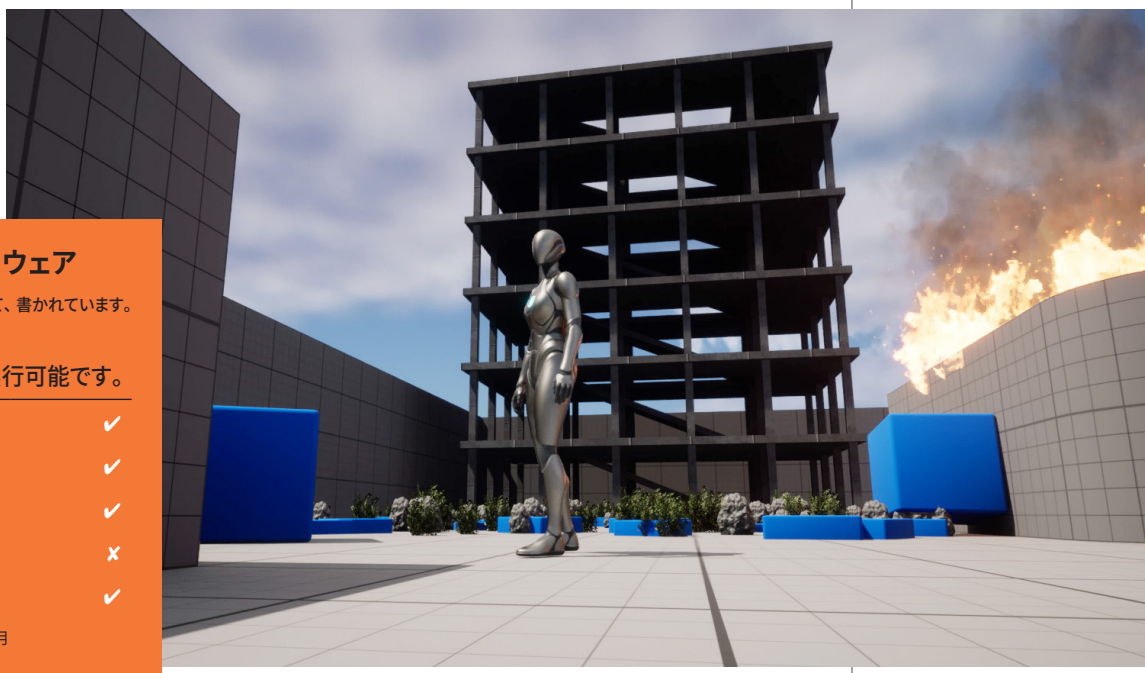
このレッスンは、**Unreal Engine 5** で実行可能です。Unreal を使ってレッスンを進めていきますが、Houdini Engine を使用して、同じアセットを Unity にインポートすることもできます。

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。

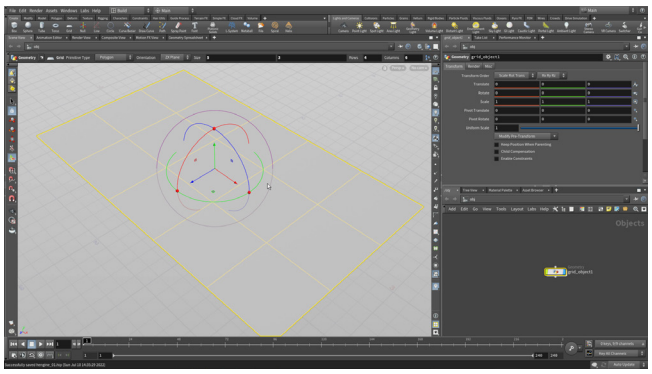
Houdini Core	✔
Houdini FX	✔
Houdini Indie	✔
Houdini Apprentice	✘
Houdini Education	✔

ドキュメントバージョン 3.0 | 2022 年 7 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。



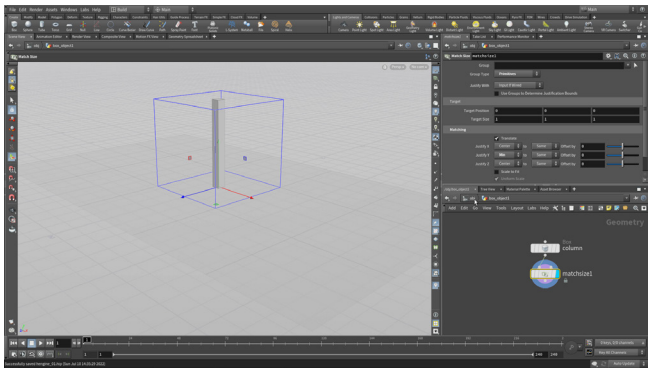
01

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。

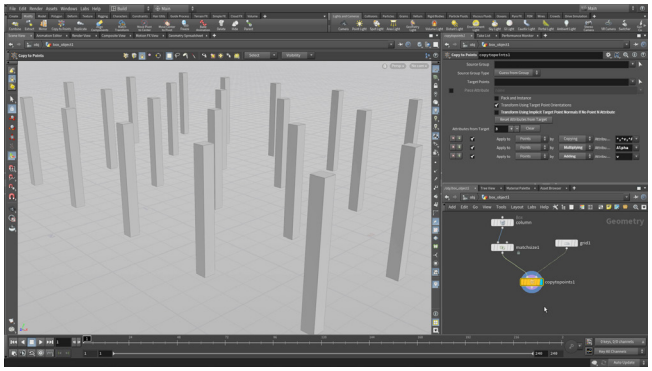


02

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。

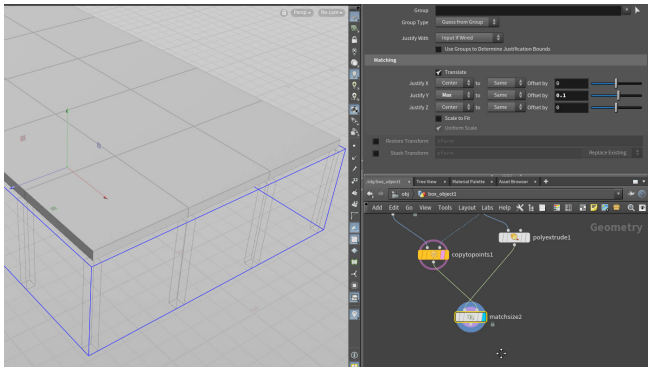


03

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。

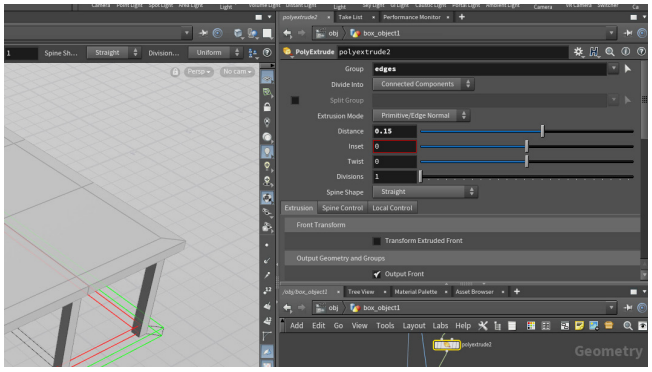


04

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。

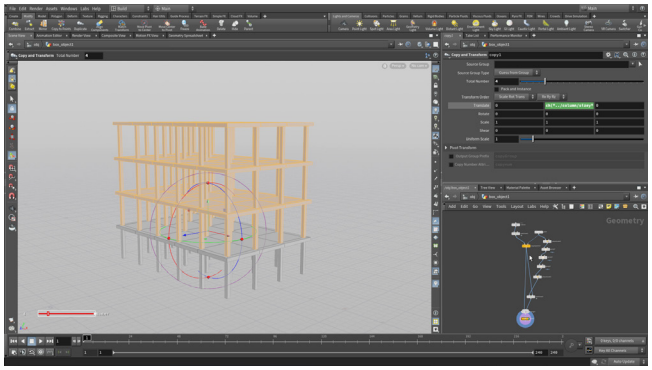


05

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。

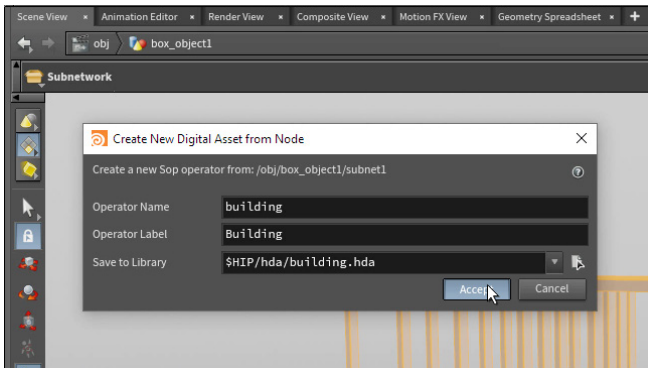


06

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。

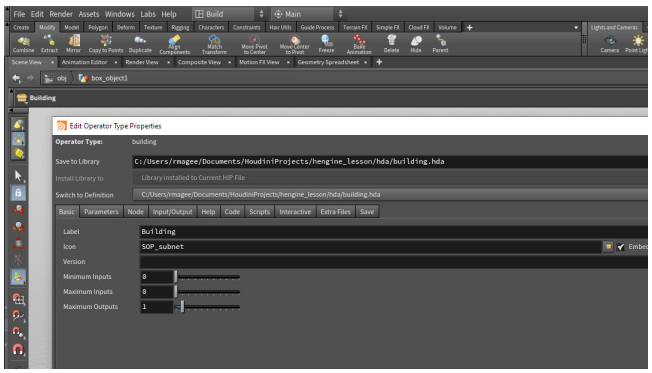
ここで、**Total Number** を 4 に増やして、3 フロア追加します。



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。

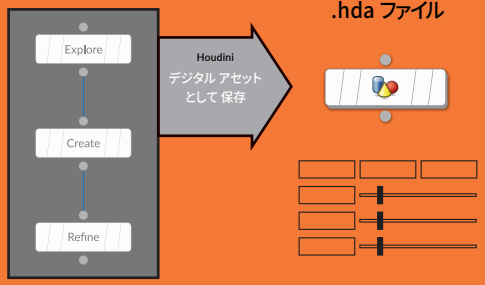
ネットワークビューのノードを building という**名前に変更**します。

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。

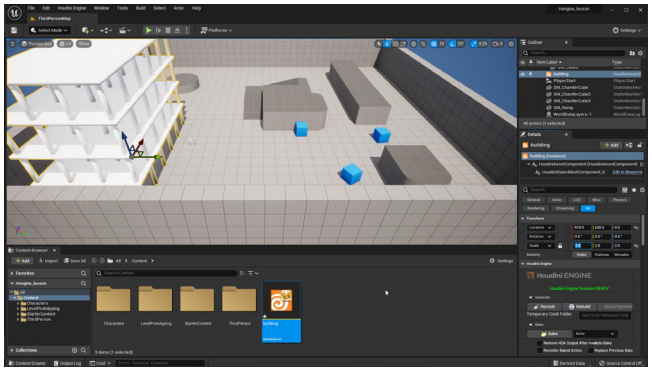
Houdini の旧バージョンで作成したアセットファイルは、**.otl** (Operator Type Library) という別の拡張子が付いていますが、どちらのタイプのファイルも同じように使用できます。



パート 2

Unreal にアセットをインポート

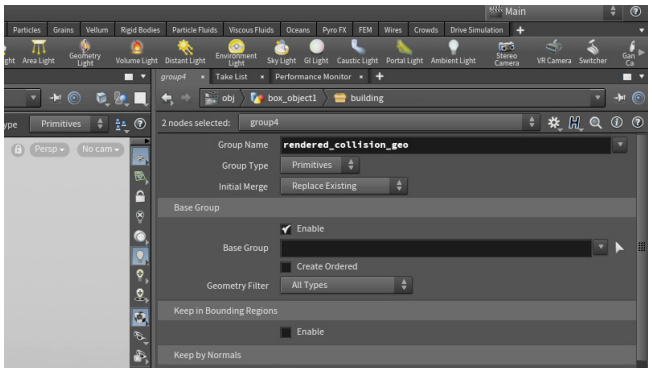
デジタルアセットファイルがディスクに保存され、Unreal Engine にインポートできる状態になりました。これが可能なのは、Houdini Engine プラグインが 2 つのアプリケーションを繋いでいるからです。Unreal にロードされる Houdini デジタルアセットは、Houdini を使用して内部的にクックされます。**このレッスンを完了するには、Houdini Engine for Unreal プラグインをインストールする必要があります（Sidefx.com/unreal をご覧ください）。*Houdini Apprentice で Houdini Engine を実行することはできません。***



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。

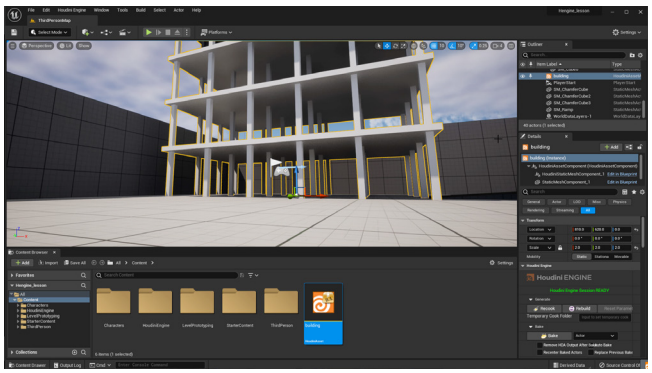


02

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。

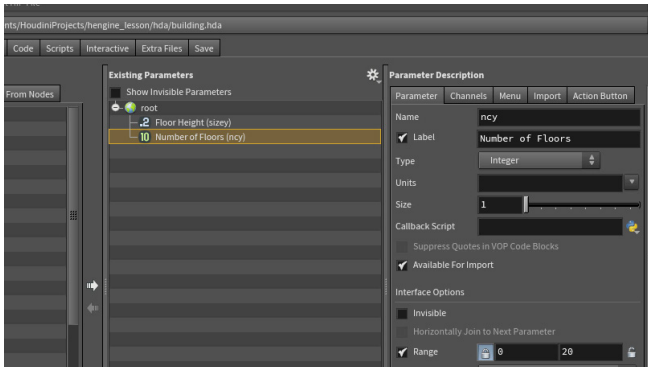
これら 2 つのノードを**コピーアンドペースト**して、新しいノードを ***column*** ノードの下に挿入します。



03

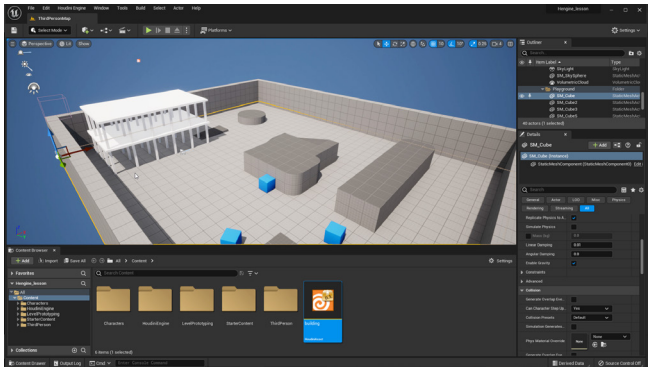
UNREAL で – ***building*** アセットの **Details** パネルの Generate セクションで、**Rebuild** を押します。**Play** を押して、シーンを確認します。

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

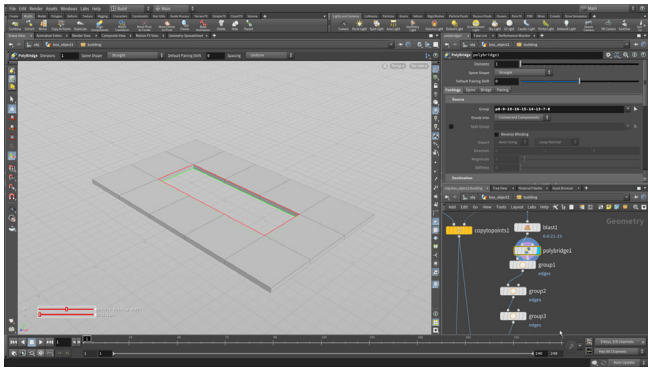
HOUDINI で – **Assets > Edit Asset Properties > Building** を選択します。Type Properties ウィンドウで Parameters タブをクリックします。***column*** ノードを選択し、**Size Y** パラメータを Parameter タブにドラッグします。それを **Floor Height** という**名前に変更**します。次に、***copy*** ノードから **Total Number** パラメータをドラッグし、名前を **Number of Floors** に変更します。**Accept** をクリックします。



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。

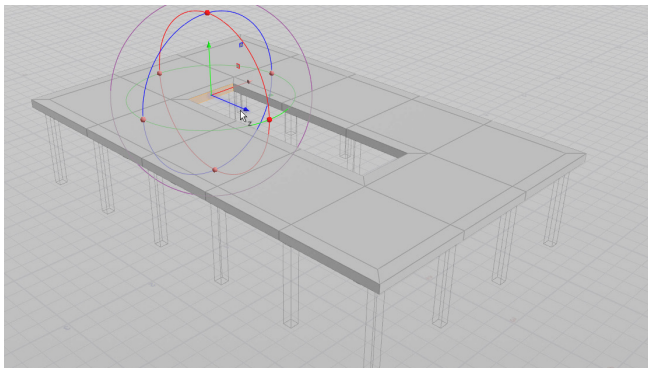
Floor Height を **1.2**、**Number of Floors** を **2** に設定します。

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。



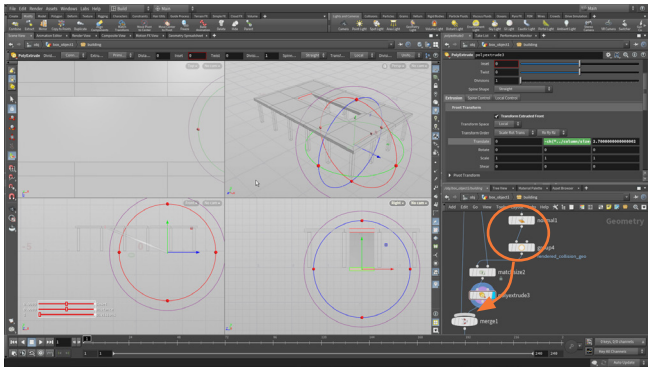
06 **HOUDINI で** – HOUDINI に戻り、床板である ***polyextrude*** ノードに **Display フラグ**を設定します。**4 を押して**プリミティブ選択にします。上部の中央にある 3 つのプリミティブを Shift を押しながら選択したら、そのままタンブルして、スラブの下部の 3 つのプリミティブを選択します。**Delete** を押します。これにより、**Blast** ノードが追加されます。

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。



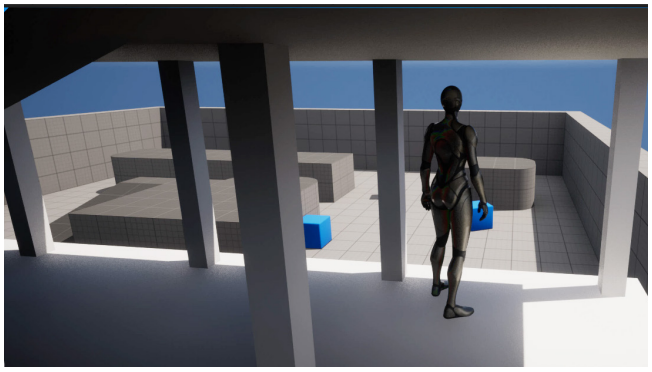
08 ***column*** ノードの **Size Y** チャンネルをコピーし、**Paste Relative Reference** で新しい ***polyextrude*** ノードの **Translate Y** にペーストします。エクスプレッションに –(マイナス)記号を追加し、0.1 を減算します。エクスプレッションは次のようになります。

```
-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。

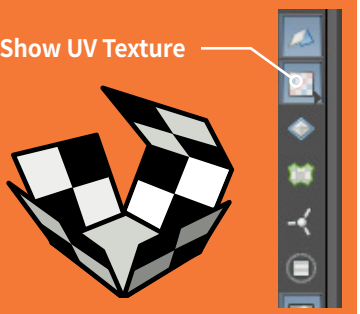


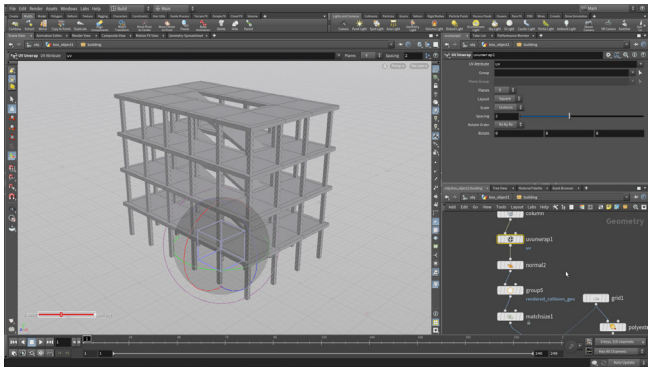
09 **UNREAL で** – **Rebuild** を押し、**Floor Height** を **1**、**Number of Floors** を **6** に設定します。**Play** を押し、斜面を上がってビルを見て回ります。

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。

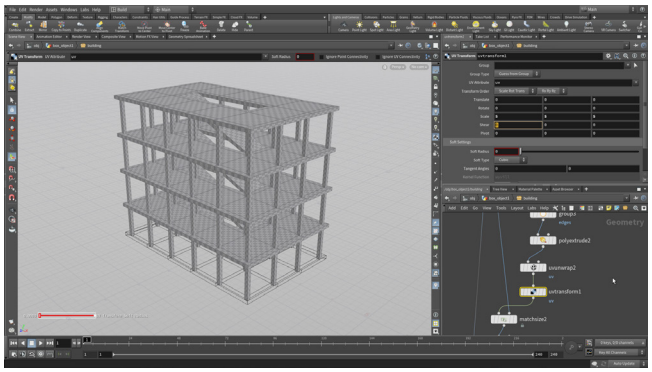
UV を配置すると、ジオメトリに UV グリッドが表示されます。グリッドを非表示にしたい場合は、**Display Options** バーの **Show UV Texture** ボタンをクリックすると、オンとオフを切り替えられます。





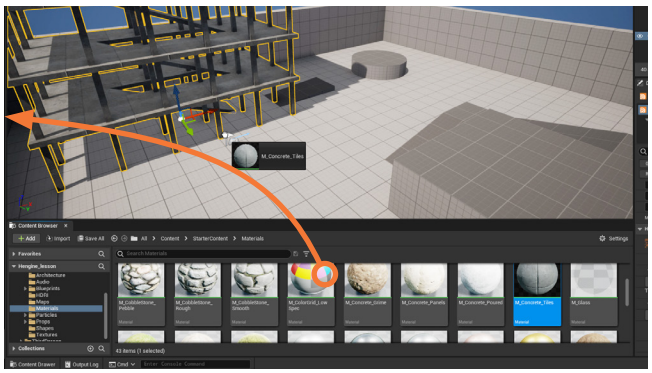
10 **HOUDINI で** – ゲームエディタで使用するには、ジオメトリに UV を追加する必要があります。ネットワークの ***column*** ノードがある部分に移動します。**Tab > UV Unwrap** を押し、***column*** ノードと ***normal*** ノードの間にノードを配置します。

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。



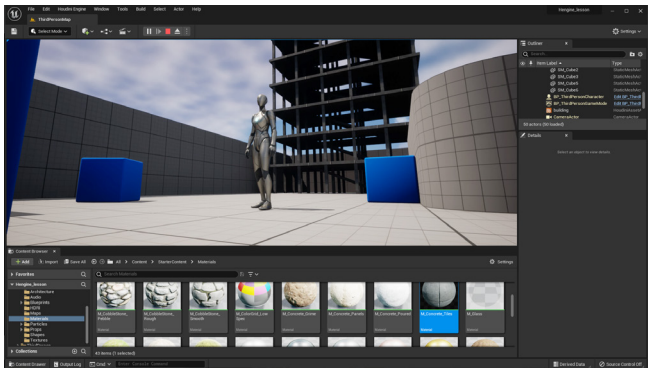
11 柱に UV グリッドが表示されます。柱と床のグリッドのスケールが一致していません。**Tab > UV Transform** を押し、***uvunwrap*** ノードと ***matchsize*** ノードの間にノードを配置します。**Scale** を **5, 5, 5** に設定します。これで、同じような UV になります。

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。

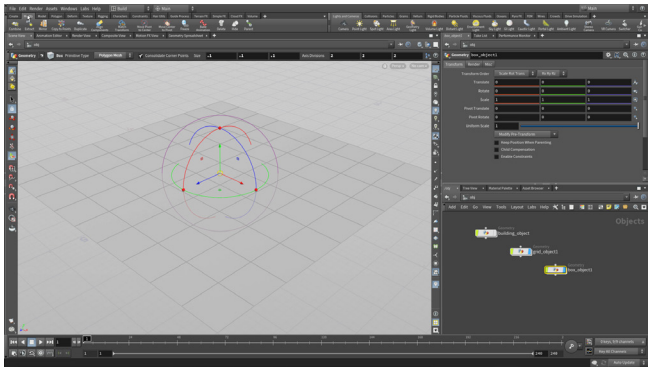


13 **Play** を押してレベルを歩き回り、テクスチャ付きのビルを探検しましょう。

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

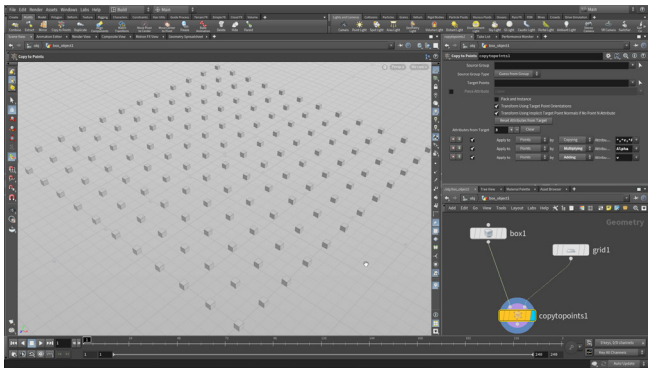
このパートでは、いくつかのボックスを別のグリッドのポイントにコピーします。次に、ポイントをランダム化してより有機的なルックにしたり、ランダムなアトリビュートを追加してボックスを回転およびスケールすることで、形状の分布にバリエーションを持たせます。このようにしてもう 1 つプロシージャルシステムを作成し、後ほど Houdini デジタルアセットにラップします。



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。

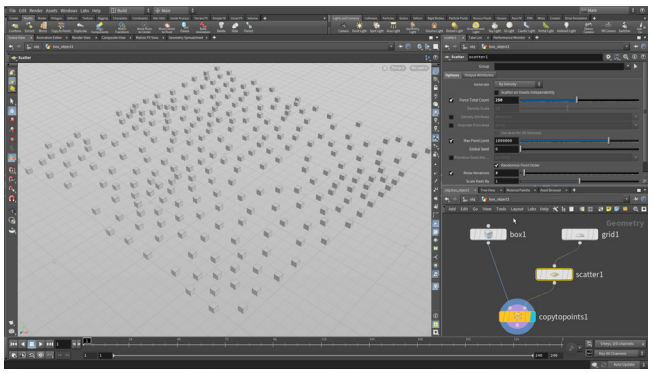
同じ Radial メニューを使用して **Create > Geometry > Box** を選択し、再度 **Enter** を押して原点に配置します。ビューポートの上部の**オペレーションコントロール**ツールバーで、**Size** を **0.1 0.1 0.1** に設定します。このジオメトリを、グリッド上のポイントにコピーしていきます。



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

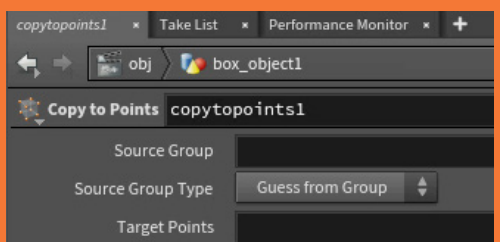
ネットワークエディタで、**Tab > Scatter** を選択します。ネットワークエディタの ***grid*** と ***copytopoints*** ノードの間に ***scatter*** ノードを配置します。すると、このノードが自動的にネットワークに接続され、ボックスが新しいポイントにコピーされます。

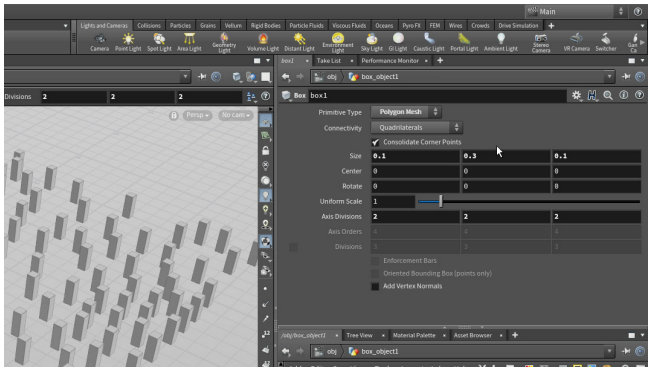
Force Total Count を **250** に設定します。**Relax Iterations** を変更し、ポイントの配置を調整します。scatter ノードを使用すると、元のグリッドポイントよりも有機的な配置にすることができます。

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。

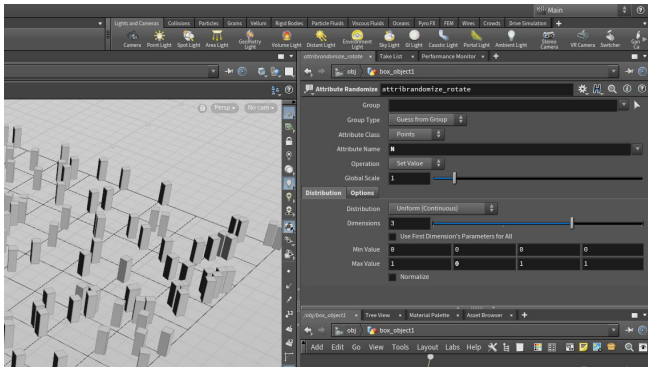




04

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。

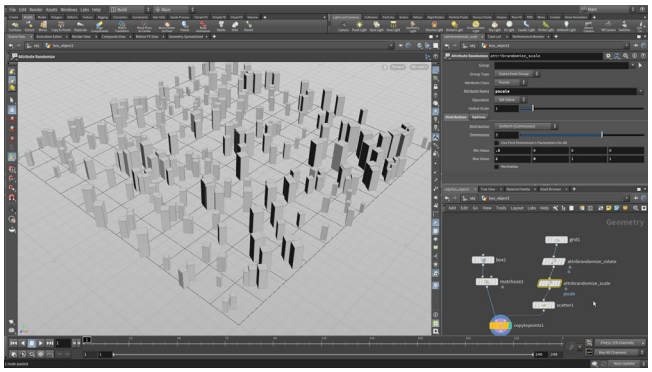
box ノードで、**Size Y** を **0.3** に設定してエクスプレッションをテストします。すべてのボックスが地面の上にあることが確認できます。



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。



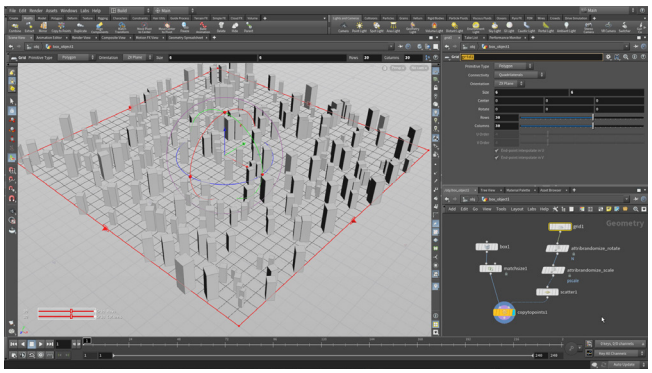
06

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

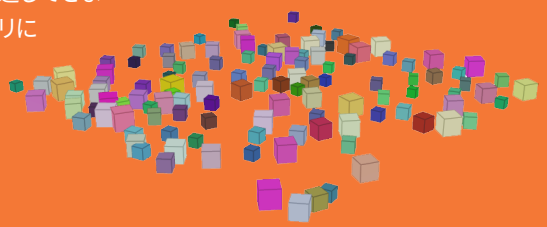
grid ノードを選択し、**Rows** と **Columns** を **30** に上げます。グリッドのポイント数が増え、さらにランダムにポイントがばら撒かれるようになります。

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。

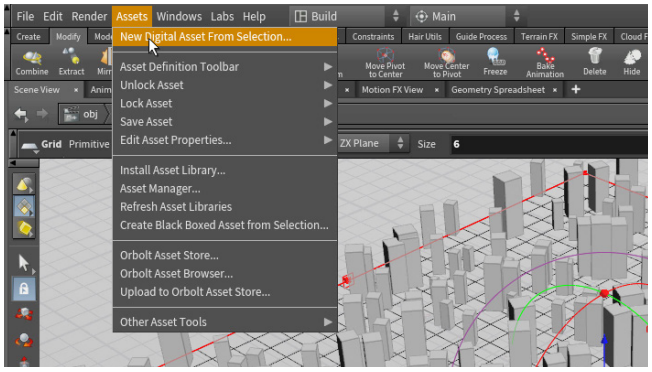
アトリビュートは最初グリッド上のポイントに割り当てられるため、ポイント数が多いほど、アトリビュート値がランダムになります。



パート 4

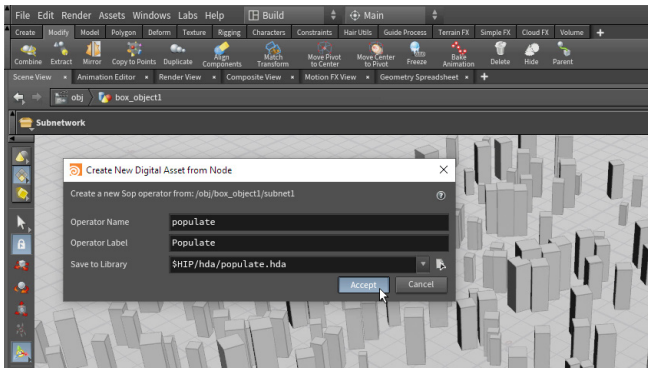
Houdini デジタルアセットをもう 1 つ作成

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。



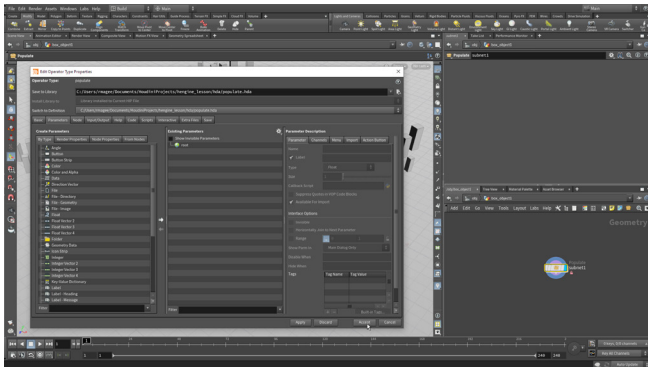
01 ネットワークエディタですべてのノードを選択します。**Assets** メニューから、**New Digital Asset From Selection** を選択します。これにより、ネットワークがサブネットワークに折り畳まれ、そのサブネットノードを使用してデジタルアセットが作成されます。

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。



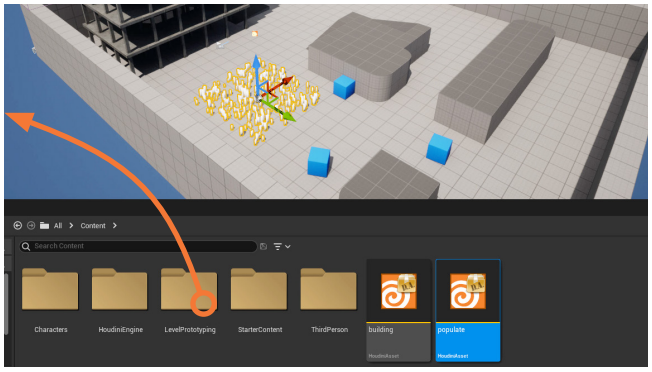
02 **Operator Name** を ***populate*** にすると、**Operator Label** が ***Populate*** に変更されます。**Save to Library** の右端のボタンをクリックします。***Locations*** サイドバーで、$HIP/ をクリックしてから hda ディレクトリをダブルクリックします。**Accept** を押したら、再度 **Accept** を押してアセットをディスクに保存します。

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。



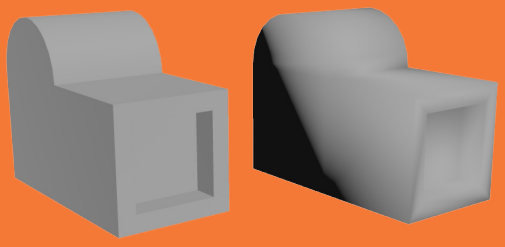
04 **UNREAL で** – Content Browser で、***Content*** ディレクトリに戻ります。**Import** をクリックし、現在のプロジェクトディレクトリの ***populate.hda*** ファイルを指定します。そのアセットを Content Browser から 3D ワークスペースにドラッグします。

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。

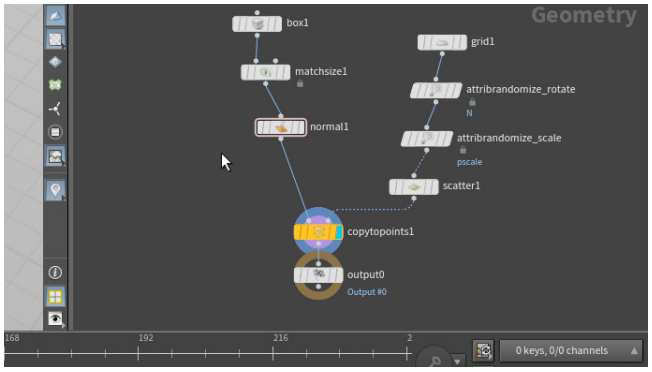
頂点法線

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。

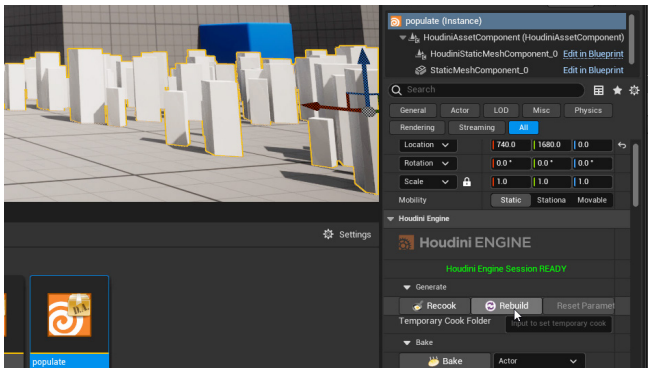


法線あり　　法線なし



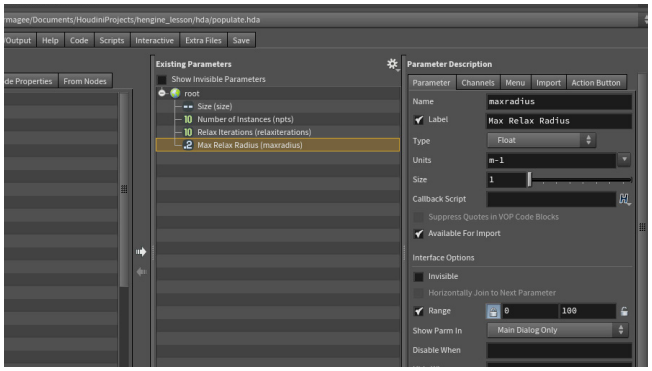
05 **HOUDINI で** – Unreal でアセットについて気になるのは、ボックスの角がくっきりしていないことです。これを修正するには、Houdini に戻り、ネットワークエディタで **Tab** を押して **norm...** と入力していき、**Normal** ツールを選択します。

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。



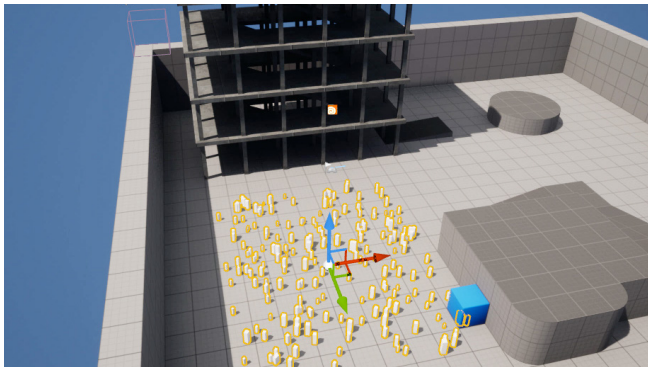
06 **UNREAL で** – Details パネルの Houdini Asset で、**Cooking Actions** セクションを開きます。**Rebuild** ボタンをクリックし、変更を確定します。これにより、Houdini 内で加えられた変更が、Unreal シーンで適切に更新されます。

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。



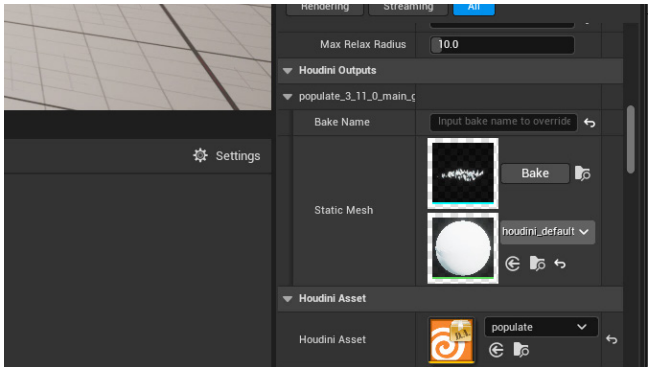
08 **UNREAL で** – **Rebuild** ボタンをクリックして変更を確定します。パラメータエディタにプロモートしたパラメータが表示されています。**Size** と **Number of Instances** を変更して、ボックスのグリッドのルックにどう影響するかを確認します。

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

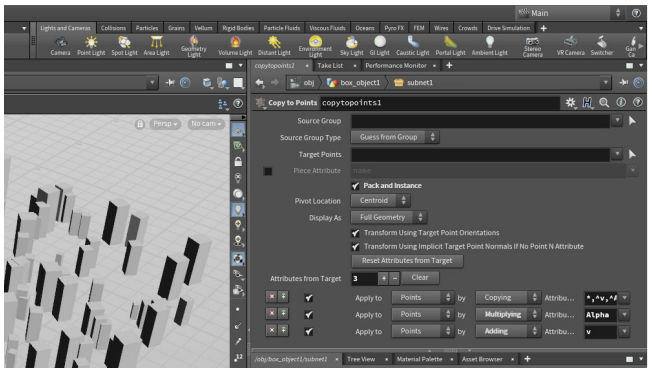
インスタンス化のセットアップ

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。



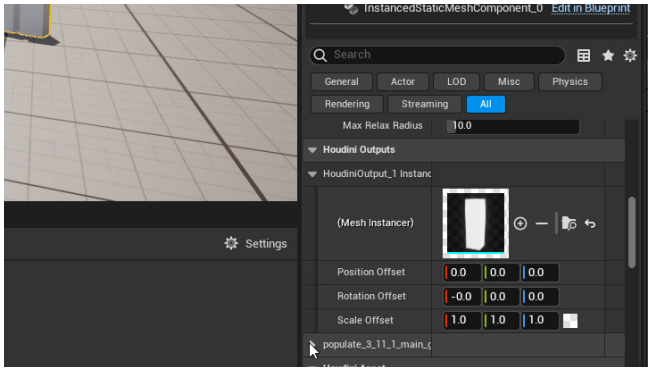
01 **UNREAL で** – Details パネルで **Houdini Outputs** を開きます。すべてのボックスが単一のメッシュとしてインポートされていることが分かります。つまり、まだインスタンス化は使用されていません。Houdini は、ボックスをポイントにコピーし、結果のジオメトリを Unreal 用に出力しています。

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

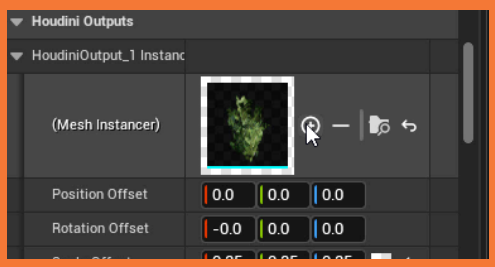
Houdini では、パックプリミティブを使用すると、ビューポート表示やレンダリング向けに効率的にインスタンスを管理できます。コピーノードに接続されているジオメトリが単一のプリミティブにパックされ、インスタンスのように扱われます。このレッスンで示されているボックスのコピーであれば、パック化を実行すると、プリミティブが 1,500 から 250 に減少します。Houdini Engine for Unreal プラグインを使用すると、パックプリミティブは Unreal インスタンスとして認識されるため、より効率的なゲームプレイが可能になります。

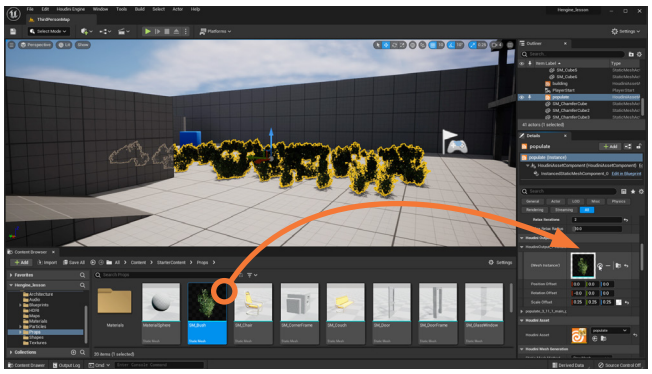
Points	2,000	Center	-0.0227511, 0.286
Primitives	1,500	Min	-3.12244,
Vertices	6,000	Max	3.07694, 0.573
Polygons	1,500	Size	6.19938, 0.573

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

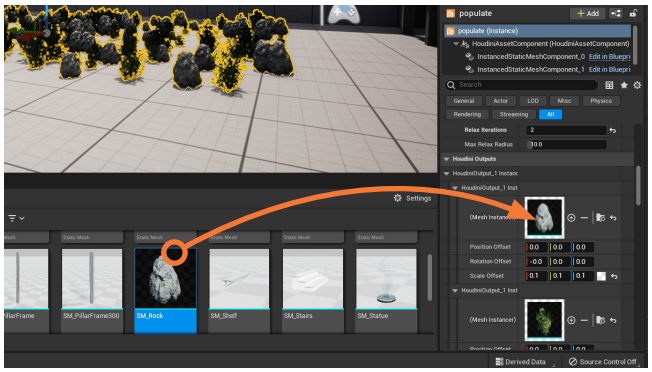
Houdini Engine プラグインがパックプリミティブを検出すると、Details タブに Houdini Output Instancer が作成されます。そこには、入力ジオメトリと、インスタンスを回転およびスケールするためのパラメータが含まれています。このインスタンスを、Unreal のコンテンツウィンドウのジオメトリと置き換え、適切なサイズに変えることができます。+（プラス）記号を使用して、インスタンス化する入力を追加すれば、同じシステム内でさらにバリエーションを作成できます。





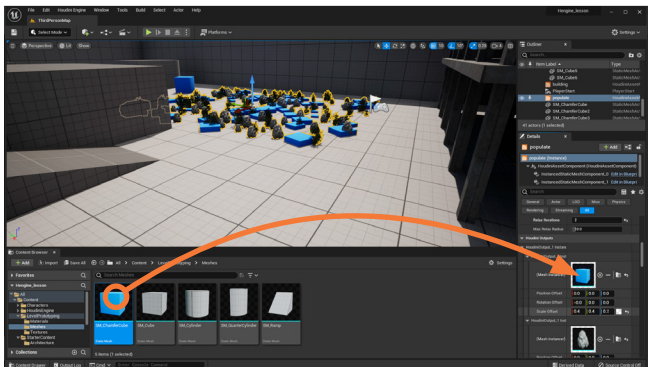
04 **Houdini Outputs** セクションに移動し、***HoudiniOutput1 Instancer*** を展開します。このボックスのインスタンスを、Unreal 内のコンテンツで置き換えます。

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。



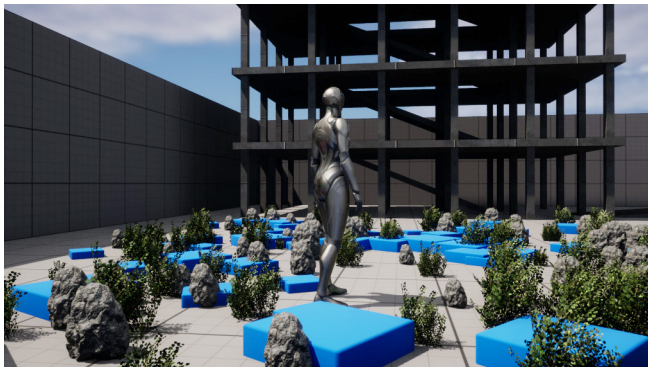
05 インスタンスオブジェクトの横にある **+（プラス）記号**をクリックします。これで 2 つ目が追加されます。***SM_Rock*** プロップを新しい **Houdini Instanced Input** 上にドラッグします。3 軸すべての **Scale Offset** を **0.1** に設定します。

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。

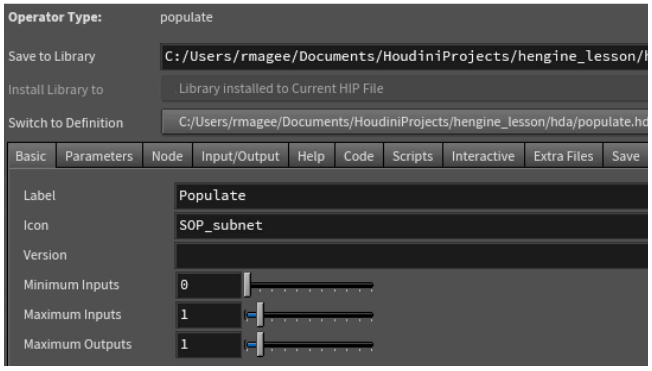


07 ***building*** アセットの **Details** パネルの Generate セクションで、**Rebuild** を押します。Bake セクションで、**Replace Preview Bake** をオンにし、**Bake** ボタンをクリックします。

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。

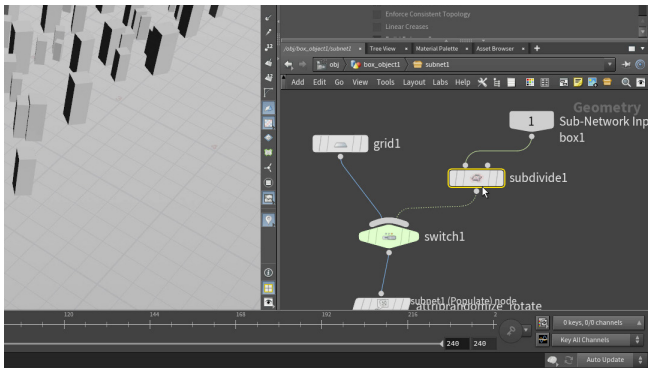
パート 6
ジオメトリを使用してアセットを駆動

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。



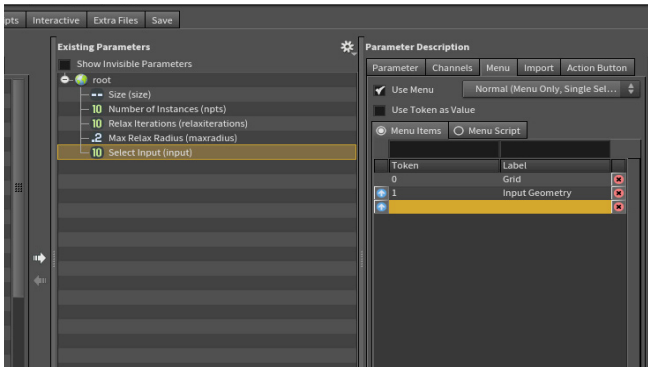
01 **HOUDINI で** – **Assets > Edit Asset Properties > Populate** を選択します。**Basic** タブで、**Maximum Inputs** を **1** に設定します。Type Properties ウィンドウで **Accept** をクリックし、変更を保存します。これによりアセット内に入力ノードが作成され、ネットワークに接続できます。後でこの入力を使用して、Unreal シーンからジオメトリを選択します。

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。

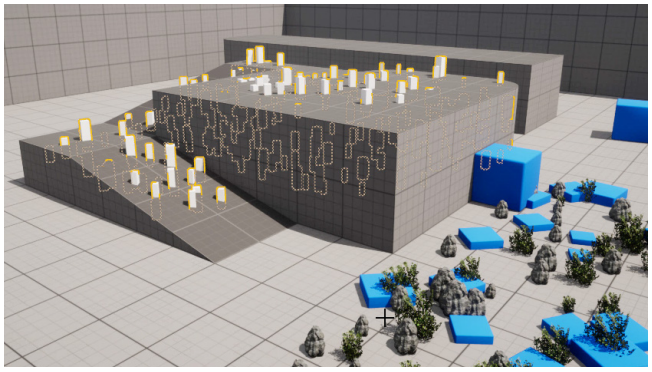
subdivide ノードに**エラー**が発生するのは、Houdini で 3 つ目の入力に何も接続されていないからです。1 つ上のレベルに戻り、**box** をアセットの入力に接続して、これがどう機能するかをプレビューしやすくします。Houdini でボックスは球に細分化されますが、最終シーンではこのようにはなりません。



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。



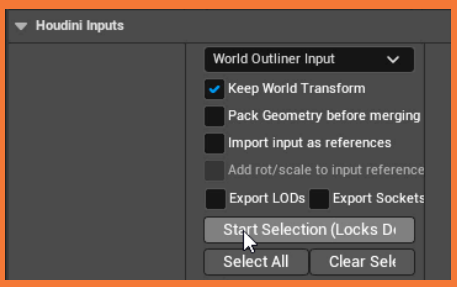
04 **UNREAL で** – **Rebuild** ボタンをクリックして変更を確定します。新しい ***populate*** アセットを前景にドラッグします。**Houdini Parameters** セクションに移動し、**Select Input** メニューから **Input Geometry** を選択します。

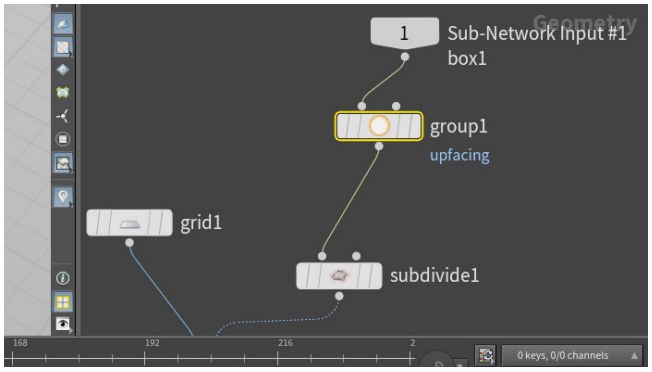
Details パネルで **Houdini Inputs** セクションに移動し、メニューから **World Outliner Input** を選択します。**Start Selection** ボタンをクリックし、3D シーンで傾斜とプラットフォームのすべてのパーツを選択します。**Use Current Selection** をクリックすると、インスタンスが細分化されたプラットフォーム内にばら撒かれます。しかし、私たちが求めているのはこれではありません。

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。

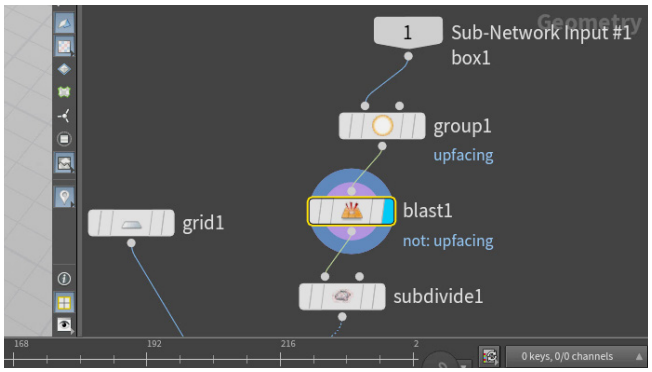
Curve Input をセットアップして、Unreal でカーブを描画できます。また、**World Outliner** からコンテンツを選択したり、Height Field に **Unreal のランドスケープ**を入力して、Houdini の新しい Terrain ツールセットで使用することもできます。





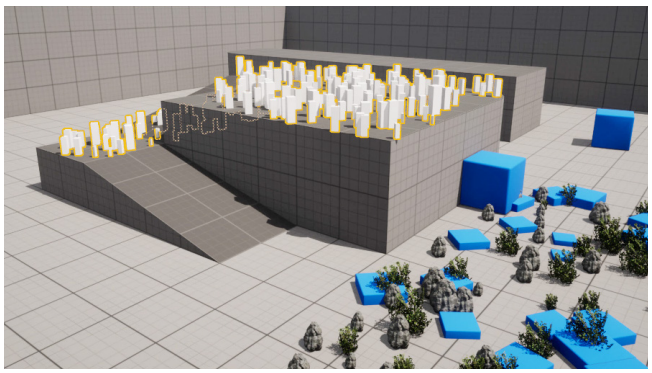
05 **HOUDINI で** – ボックスの上部のフェースを分離して、ポイントがコピーされる場所を制限します。ネットワークエディタで、***input*** ノードの後に **Group** ノードを挿入します。**Group Name** を **upfacing** に設定します。**Base Group** で、**Enable** を **オフ**にします。**Keep by Normals** で、**Enable** を**オン**にします。**Direction** を **0, 1, 0**、**Spread Angle** を **0** に設定します。

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。

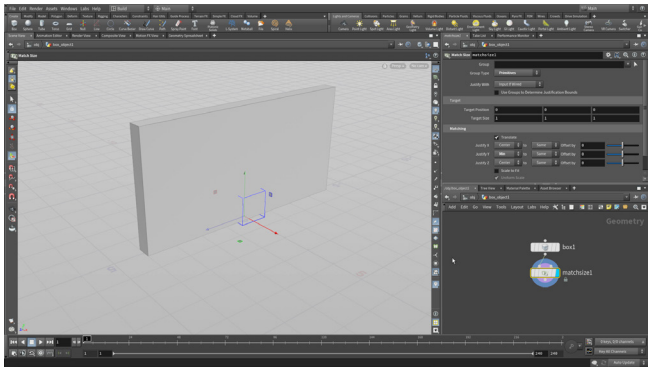
Content Browser で、**StarterContent > Particles** を開きます。***P_Fire*** プロップを **Houdini Instanced Input** 上にドラッグします。**Rotate Y** を **90** に設定します。**Play** を押して、レベルをテストします。

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

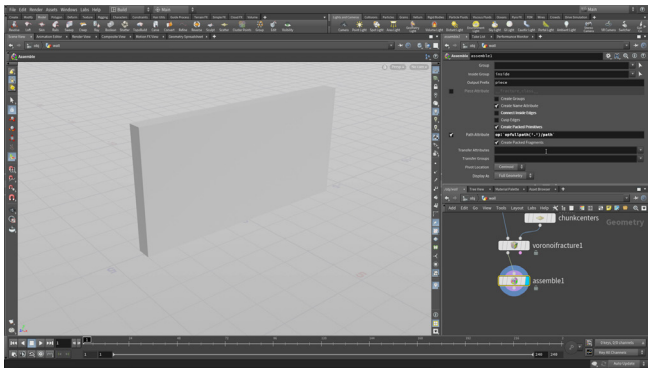
このパートでは、壁を作成し、リジッドボディダイナミクスを使用してそれを粉砕します。このシステムを FBX にエクスポートしたら、ゲームで使用するために Unreal にインポートします。Houdini から Unreal にビジュアルエフェクトを取り込む方法を、簡単な例で説明します。



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。



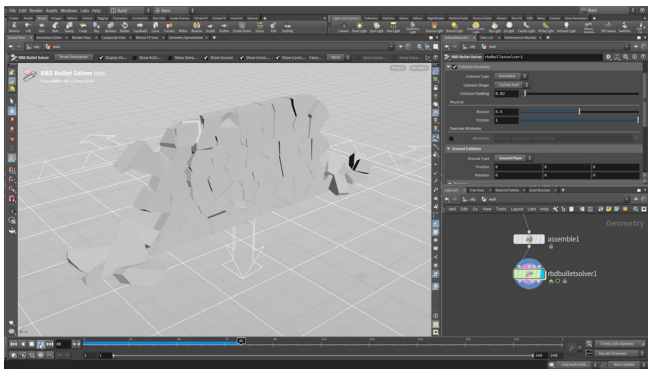
02

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。

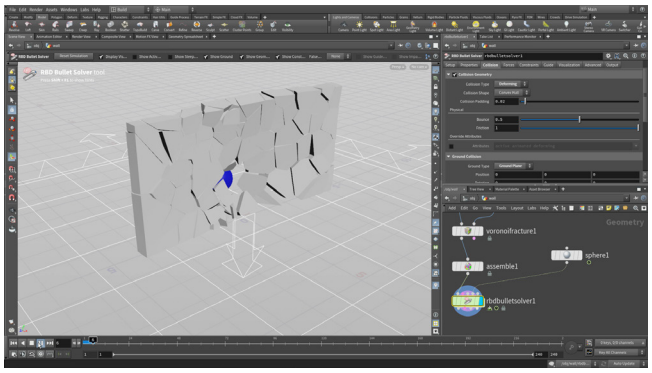
```
op:`opfullpath('.')/path'
```



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。



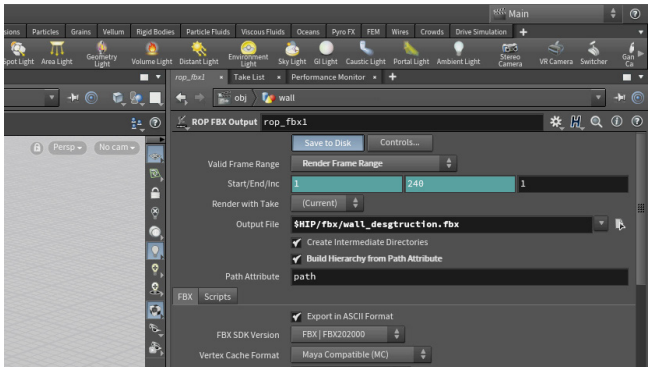
04

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）。



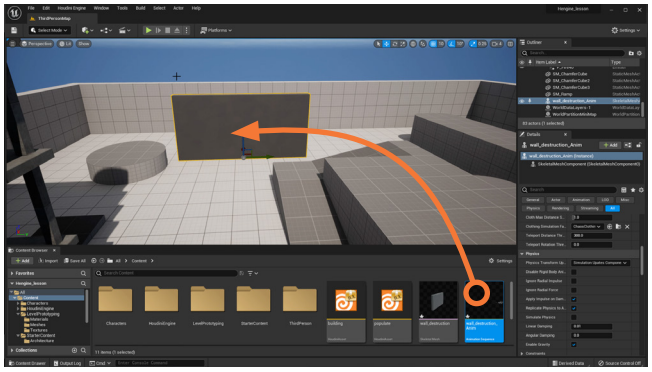
05

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。