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。

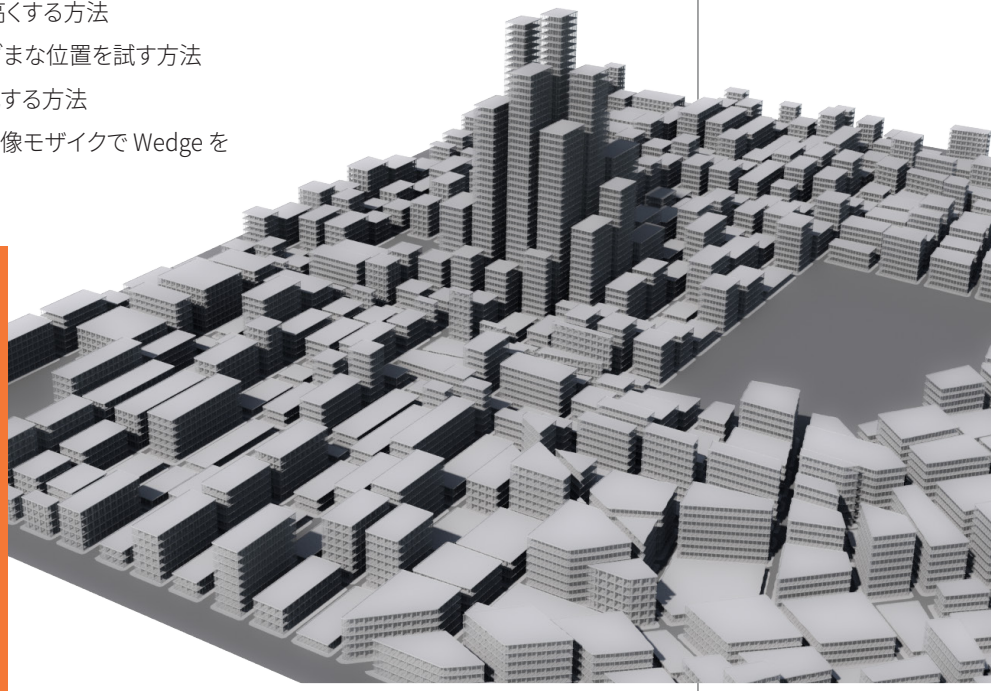
注：このレッスンでは、Image Magick を使用します。このアプリケーションがコンピュータにインストールされていることを確認してください。

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 2.0 | 2022 年 10 月

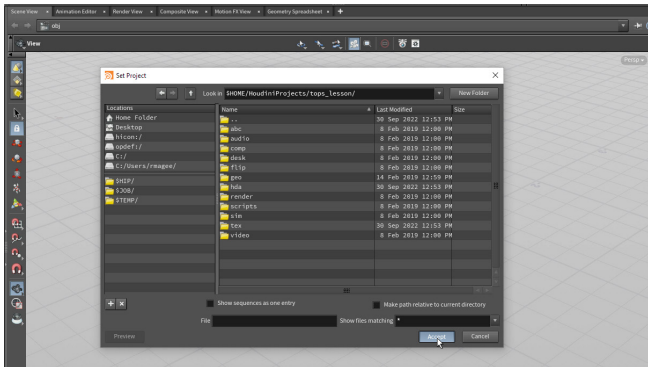
パート 1

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

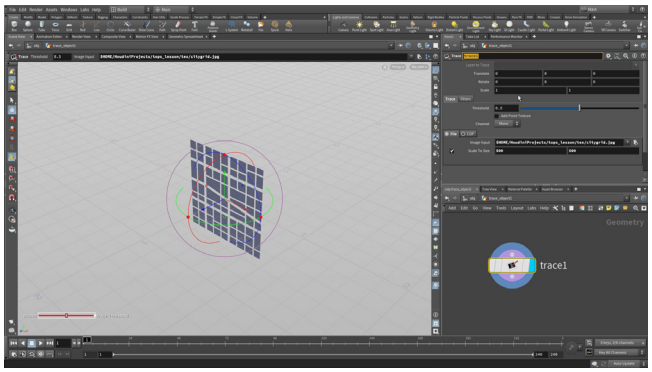
SideFX.com の **Foundations チュートリアル**のページ（このチュートリアルを入手した場所）から、***tops_lesson*** ディレクトリをダウンロードします。**home** または **documents** ディレクトリにある **Houdini Projects** ディレクトリに配置してください。



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。

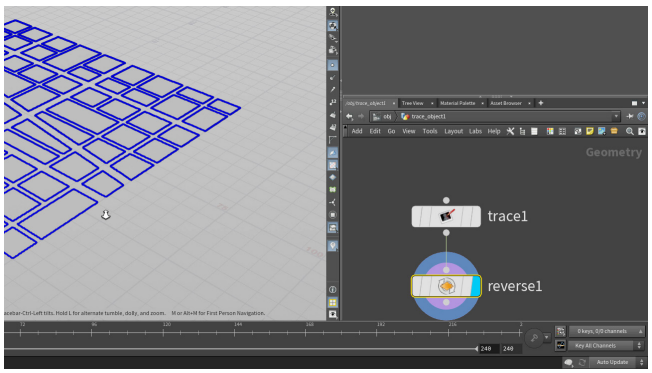
File > Save As... を選択します。新しい ***tops_lesson*** ディレクトリが表示されるはずです。表示されない場合は、左側の列で **$JOB** をクリックします。ファイル名を ***city_01.hip*** に設定し、**Accept** をクリックして保存します。



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。



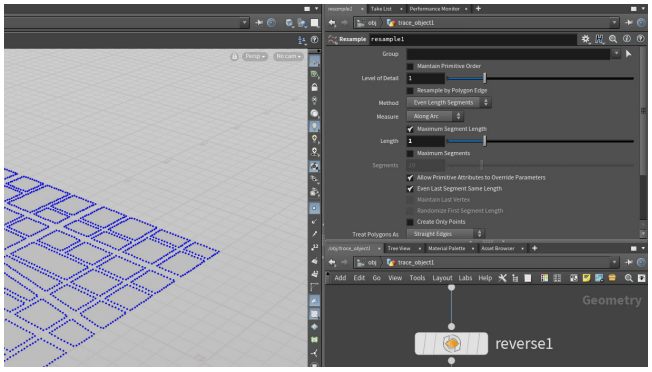
03

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。

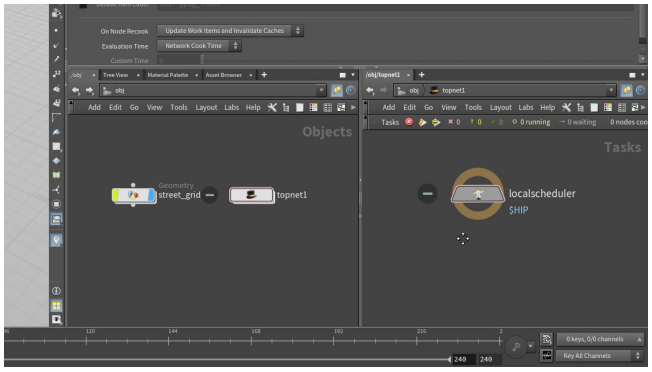
resample ノードの出力を **RMB クリック**し、**Null** を選択します。チェーンの終端に Null ノードをクリックして配置します。**Display フラグ**を設定し、名前を ***CITYBLOCKS_OUT*** に変更します。

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

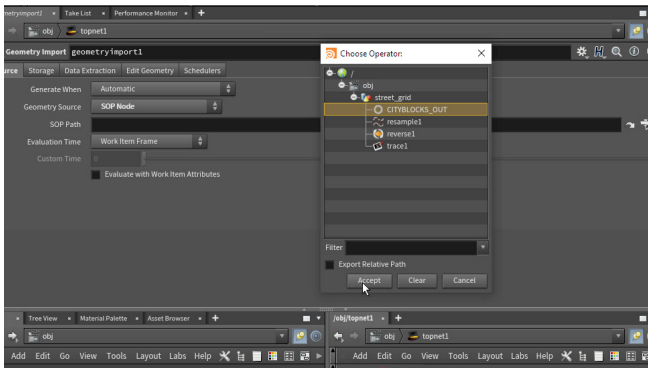
都市グリッドを作成できたら、さまざまな都市ブロックを個別のワークアイテムに分割します。これで、ブロックごとにビルを生成できるようになります。各ブロックを保存するための TOP ネットワークをセットアップします。同時に、選択したワークアイテムを視覚化し、望みどおりの結果になっていることを確認するためのオブジェクトも作成します。



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。

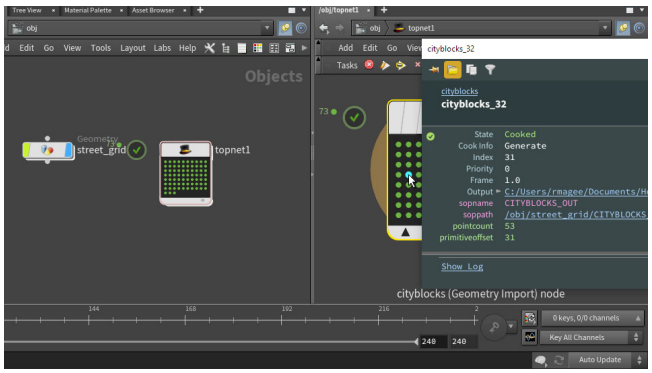
ネットワークビューの右上にある小さい**矢印**をクリックします。メニューから **Split Pane Left/Right** を選択します。左側のネットワークビューで**ピン**アイコンをクリックします。右側のネットワークビューはすでにピン留めされています。右側のネットワークビューで、***topnet*** を**ダブルクリック**して中に入ります。これで、両方のネットワークを使用して都市を構築できるようになりました。



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

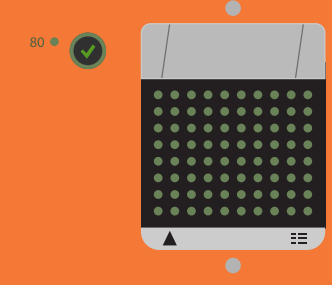
geometryimport ノードの名前を ***cityblocks*** に変更し、上部の**タスクバー**で **Cook Selected Node** ボタンをクリックします。または、このノードを選択して **Shift + G** を押しても、クックすることができます。ノードがクックされるとき、ワークアイテムを表す小さいドットが表示されます。

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。

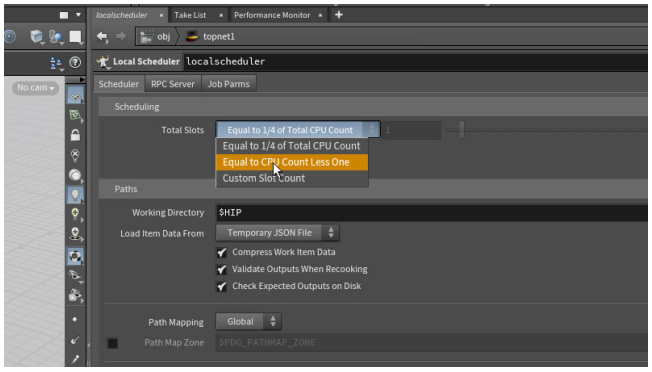
すべてのワークアイテムのクックが完了すると、ノードにチェックマークが付き、完了したワークアイテムが緑色になります。



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。

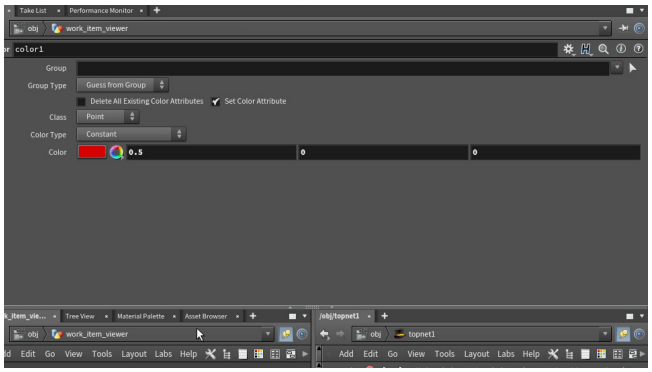


04

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。

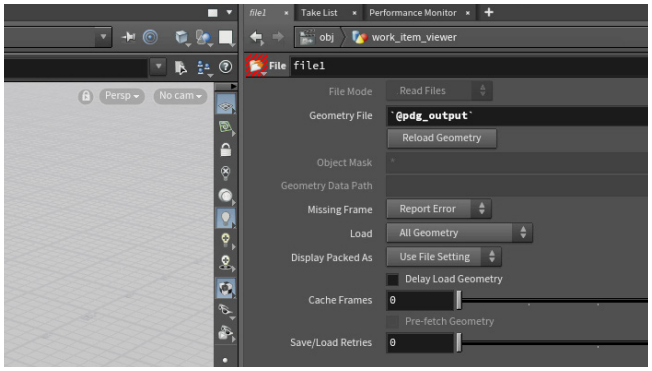
File ノードは通常、ディスク上のファイルを指します。まずはこの方法でファイルを見つけ、その後、別の方法を使ってワークアイテムを TOP から直接取得します。



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。

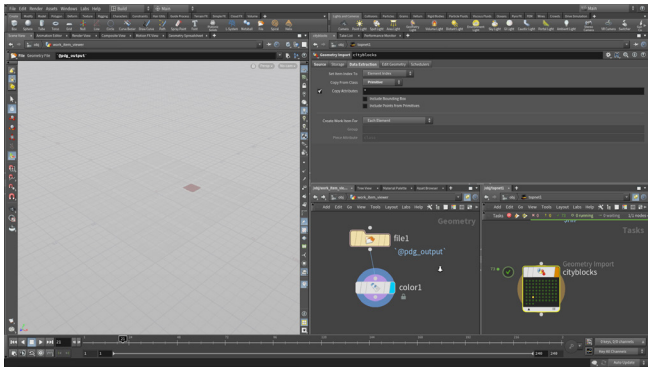


06

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

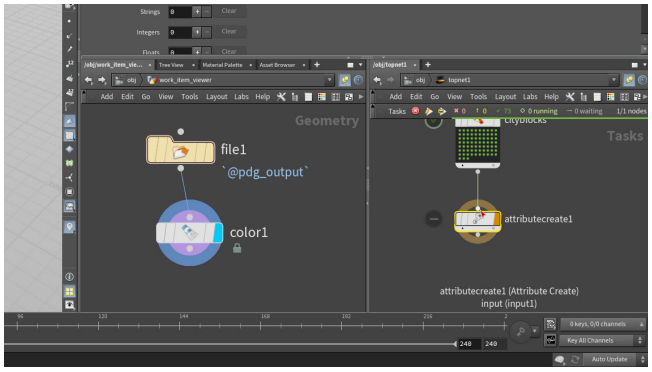
ジオメトリネットワークビューの **Active Work Item** メニューで、異なるワークアイテムを選択し、Scene View でそれらを選択します。赤色が選択したジオメトリに移動します。そのワークアイテムに関連するジオメトリがビューポートに表示されます。topnet では、黄色にハイライトされた ***cityblocks*** TOP ノード上にドットが表示されます。

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。



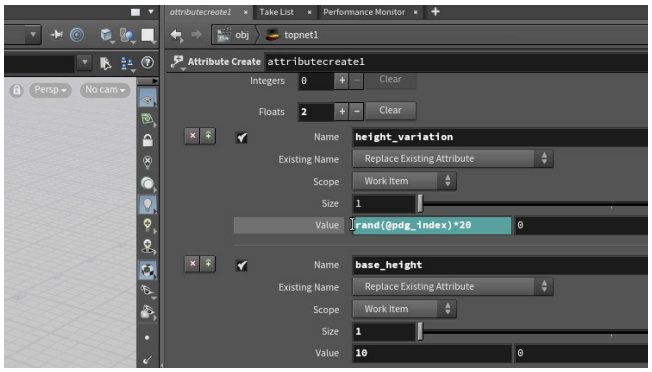
01

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。

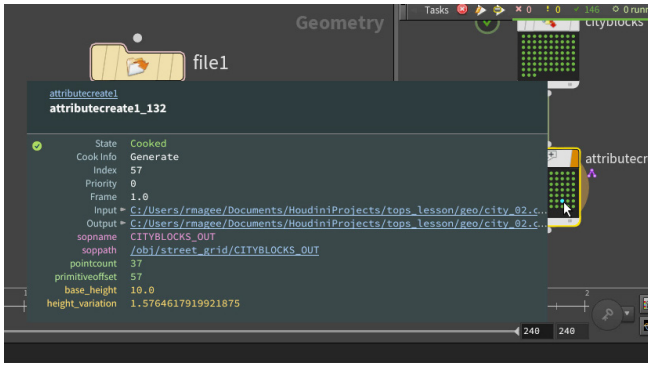


02

+（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。

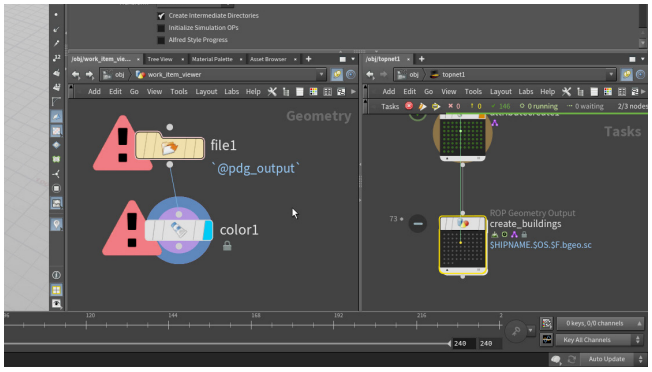
アトリビュートは、パイプラインに対して重要な情報を提供する重要な手段であるため、適切にセットアップすることが大切です。

```
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。

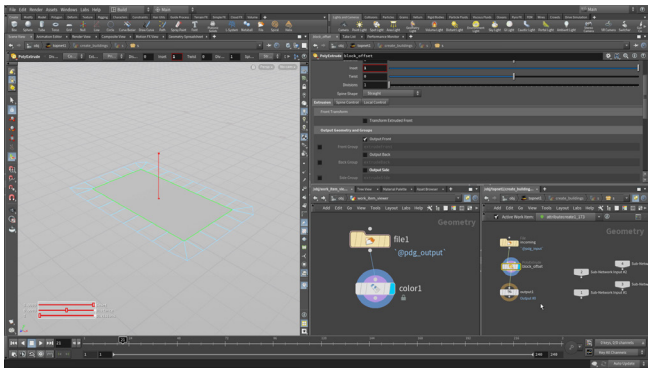


01

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。

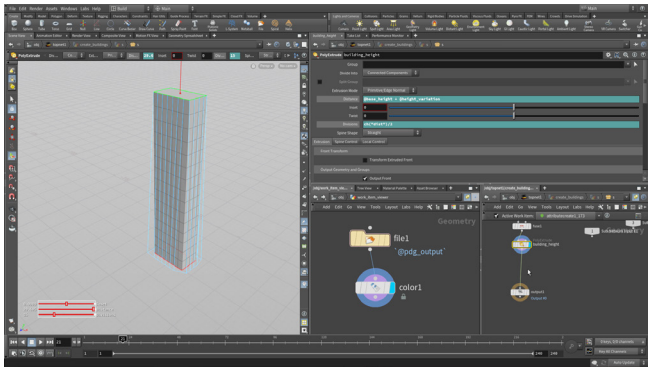


02

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

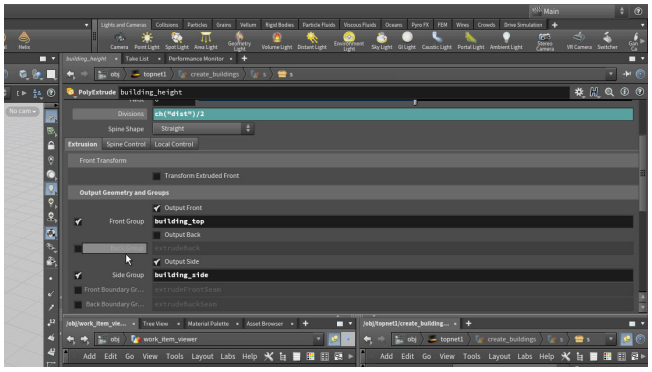
fuse の後に、2 個目の **PolyExtrude** ノードを作成します。名前を ***building_height*** に変更し、**Distance** を次のように設定します。

```
@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。

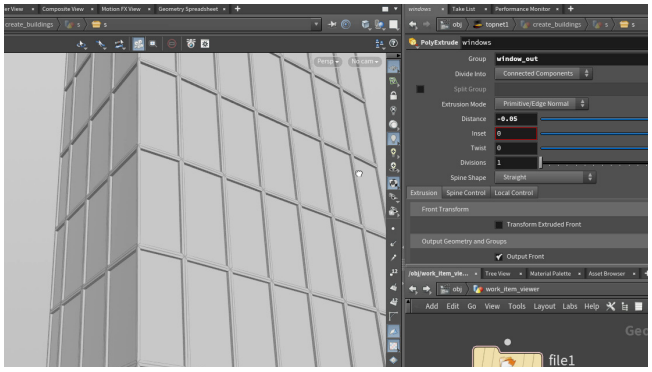
```
ch("dist")/2
```

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。

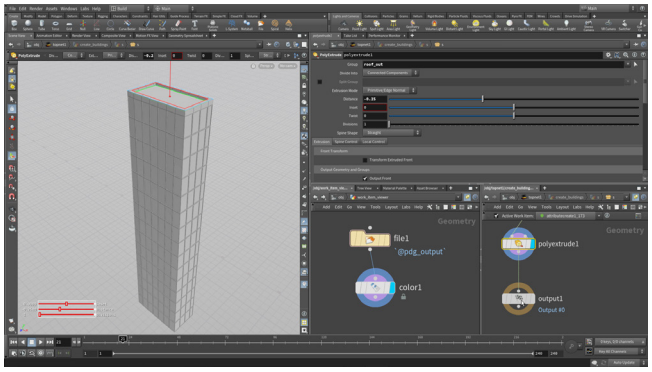


05

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

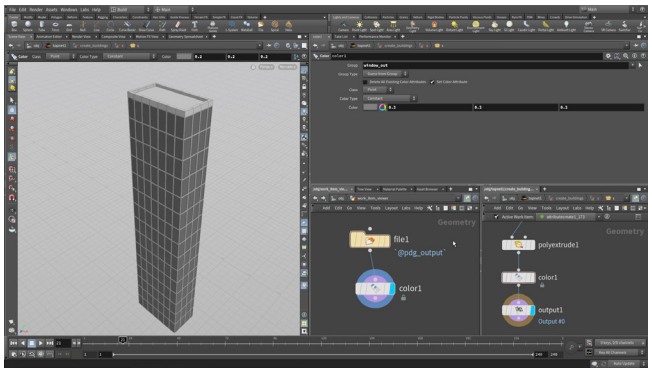
新しい **Poly Extrude** を追加し、名前を ***extend_roof*** に変更します。次のように設定します。

- **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。

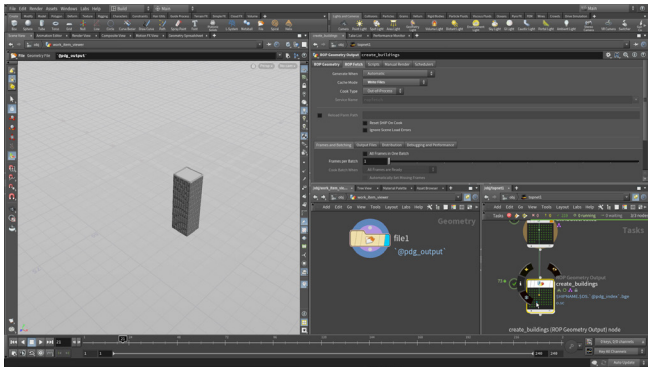


07

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。

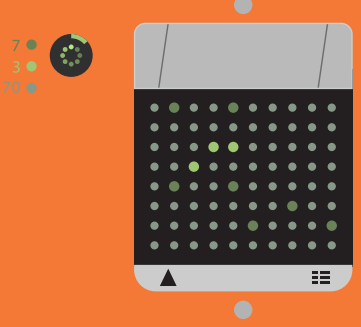
シーンファイルを**保存**したら、***create_buildings*** TOP を選択し、**Shift + G** を押してクックします。

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。

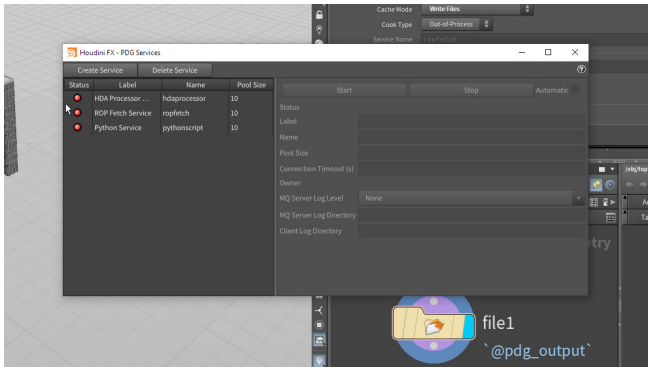
オブジェクトレベルの TOP ネットワークにも、ネットワークのすべてのタスクについてワークアイテムの進捗状況が表示されます。



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。

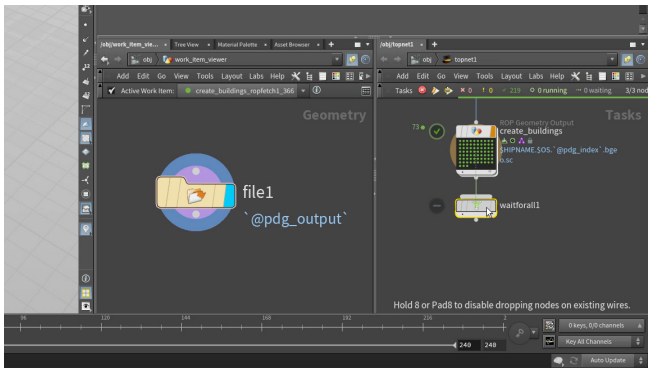


01

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。

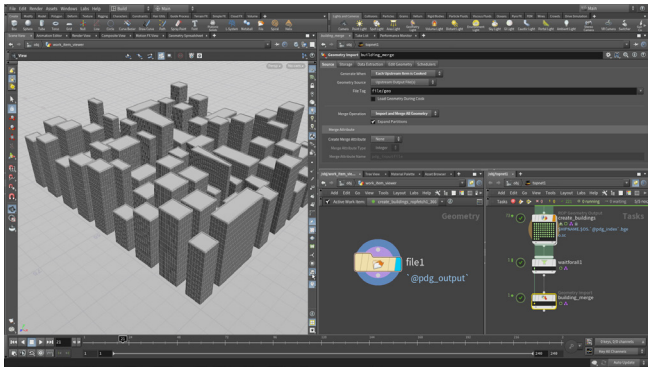
Services ウィンドウを閉じたら、***create_buildings*** TOP の **ROP Fetch** タブに移動して、**Cook Type** を **Services** に設定します。これで TOP ノードの処理速度が向上するはずです。

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

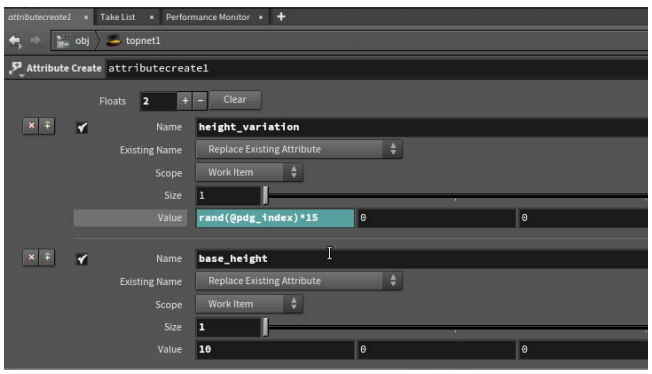
Wait for All TOP ノードを追加します。このノードは、ROP Geometry ノードによって生成されたすべてのワークアイテムを受け取り、それらを結合して単一のワークアイテムにします。このノードはまた、上流のすべてのワークアイテムのクックが完了するまで、下流の TOP がクックされないようにします。



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。



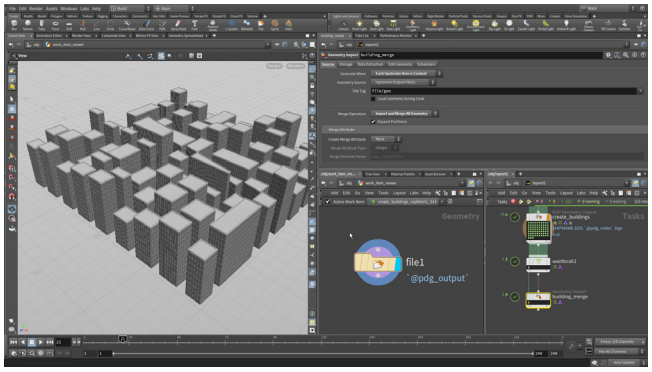
04

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

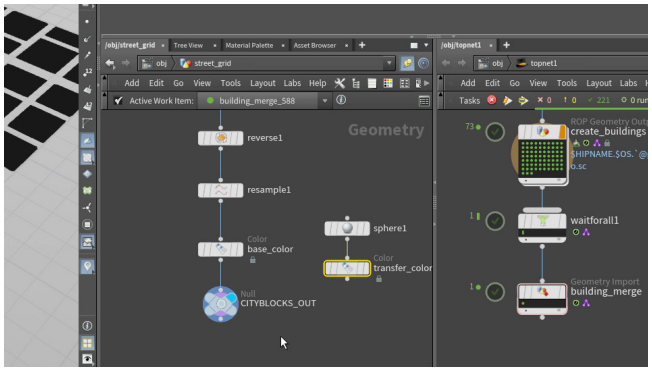
これで、ビルの高さが全体的に少し低くなります

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。



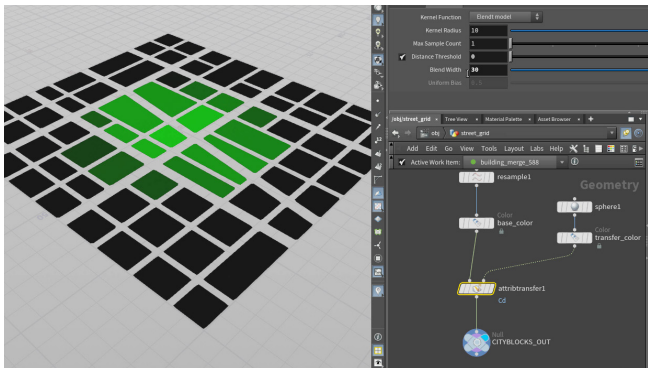
06

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。



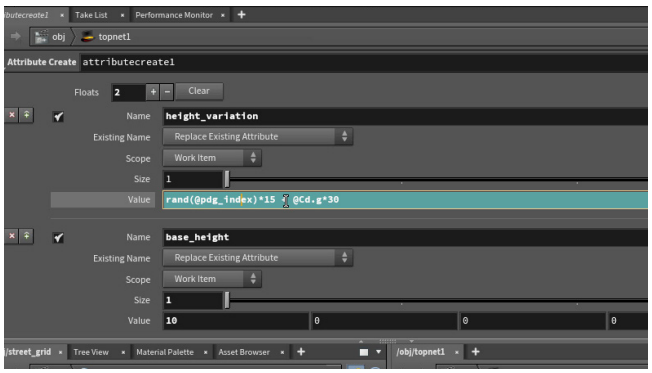
07

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

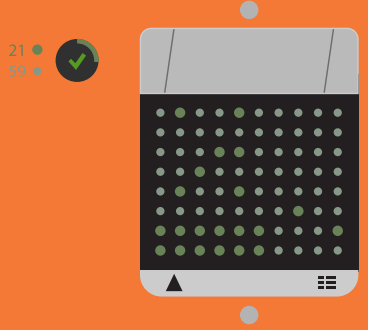
TOP ネットワークに戻り、***attributecreate*** を選択して、***height_variation*** アトリビュートを次のように編集します。

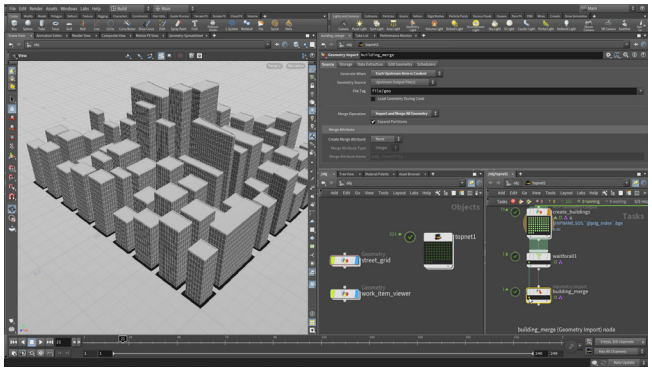
- **Value** を `rand(@pdg_index) * 15 + @Cd.g*30` にする

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。

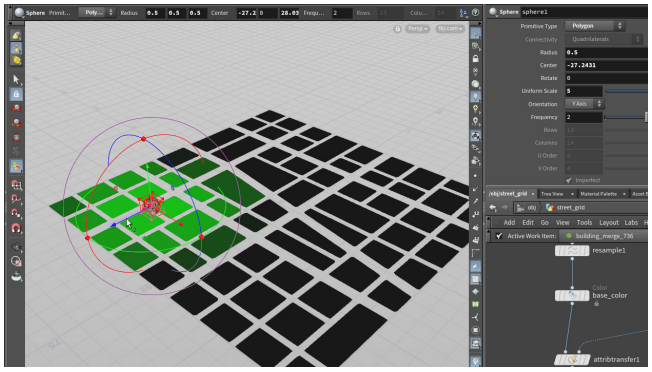




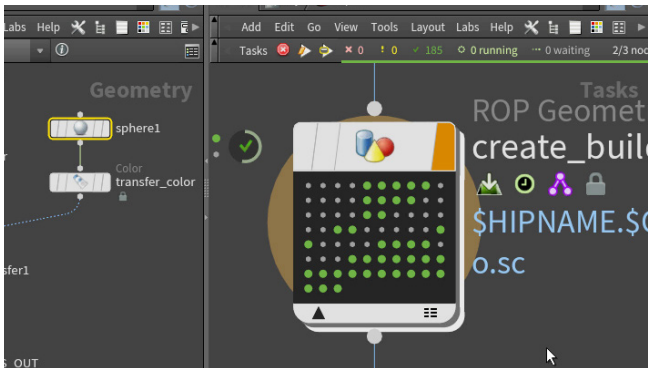
09 ***building_merge*** ノードを **RMB クリック**し、メニューから **Dirty This Node** を選択します。または、このノードを選択して **Shift+ D** を押しても、Dirty（変更あり）にできます。

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。

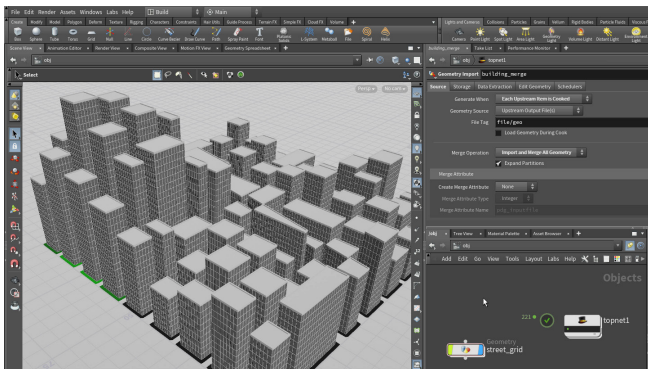
もう一方のネットワークビューで、オブジェクトレベルに戻ります。都市が更新され、ビューポートで都心部を確認できるはずです。



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。

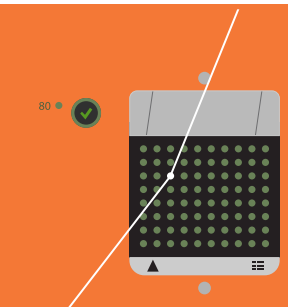


12 ***building_merge*** ノードを **RMB クリック**し、**Cook Selected Node** を選択すると、Dirty（変更あり）のワークアイテムのみが更新されます。ワークアイテムをクリックして、新しい都心部を表示します。このようにすると、ワークフローを開発する際、TOP の依存関係グラフを効率的に動作させることができます。

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。

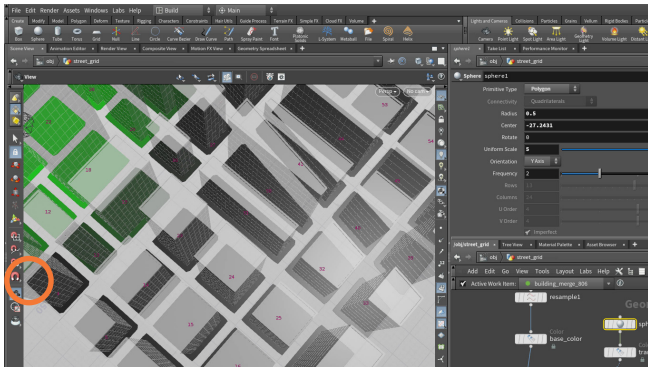
これは依存関係のマッピングであり、グラフを効率的に動作させるのに役立ちます。なぜなら、**Shift + D** でノード上のすべてのワークアイテムを明示的に Dirty（変更あり）にしない限り、Dirty（変更あり）のワークアイテムのみが処理されるからです



パート 6

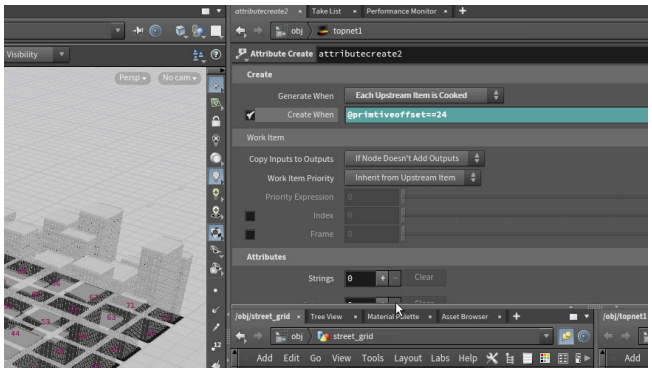
1 つのビルを分離する

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。

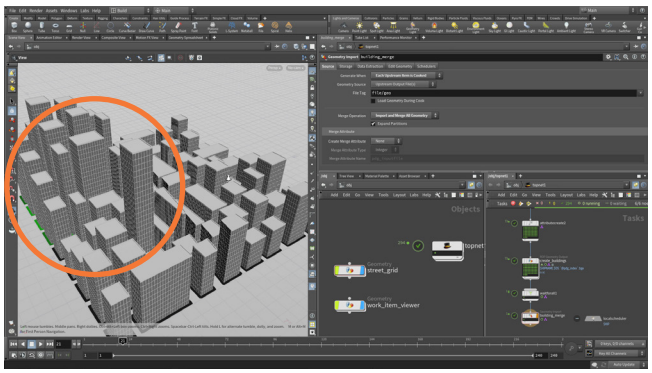
現在のセットアップのランダム性に依存する代わりに、この都市ブロックに特定のパラメータを設定して、望みどおりの正確な高さが得られるようにしましょう。すべてをランダムに任せずに、TOP ネットワーク内でアーティスティックな制御ができるようになります。



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。

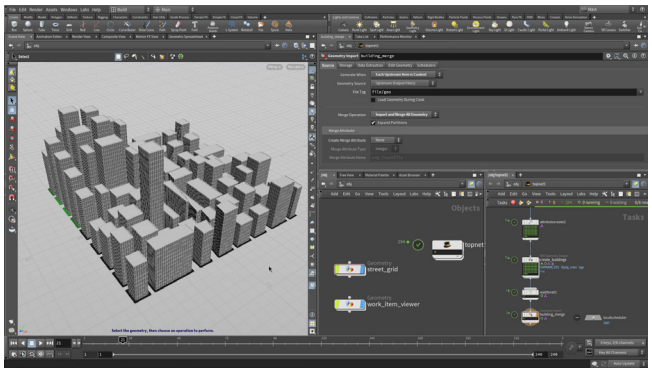
@primitiveoffset==24

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。



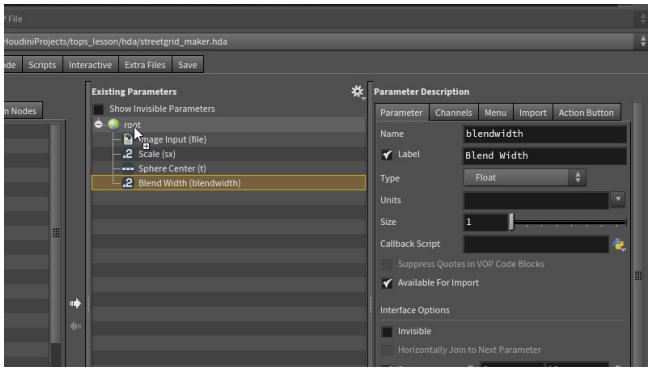
04 ***building_merge*** ノードのワークアイテムをクリックして、新しいビルがそびえ立つ都市景観を表示します。これで、ネットワークを再クックしてもブロック 24 のビルは更新されず、明示的に設定された状態に保たれるようになります。

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

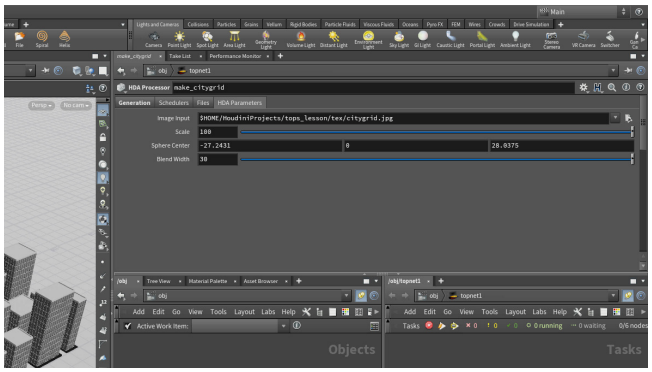
street_grid オブジェクト内で球の位置を Wedge 化するには、ネットワークを Houdini Digital Asset（HDA）に変換し、HDA Processor TOP ノードを介して実行する必要があります。これらのアセットをロードしてシステムに追加できるようになると、個別のツールから複雑なシステムを作成し、ツールを必要に応じて更新および適応することができます。



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。

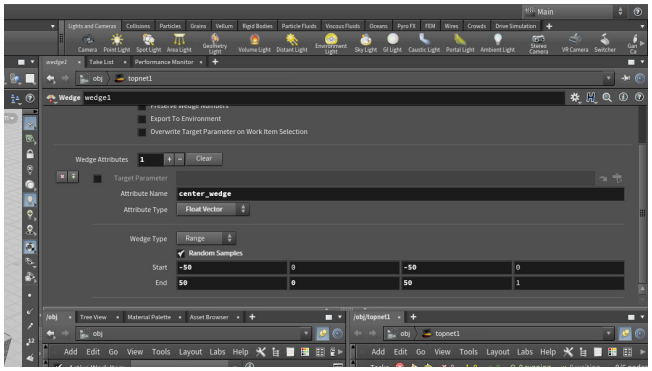


02

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

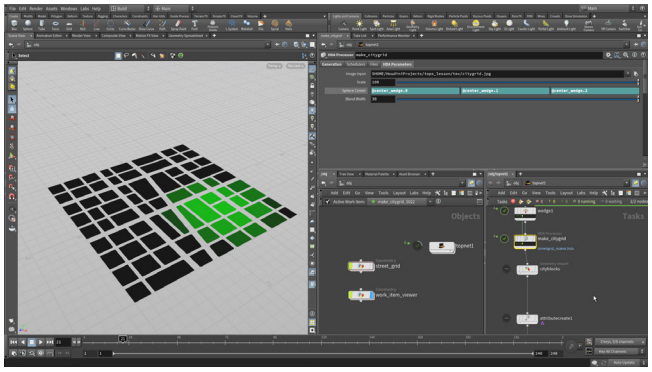
Wedge Attributes の横にある **+（プラス）記号**をクリックし、最初のアトリビュートの **Attribute Name** を ***center_wedge*** に設定したら、次のように設定します。

- **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。



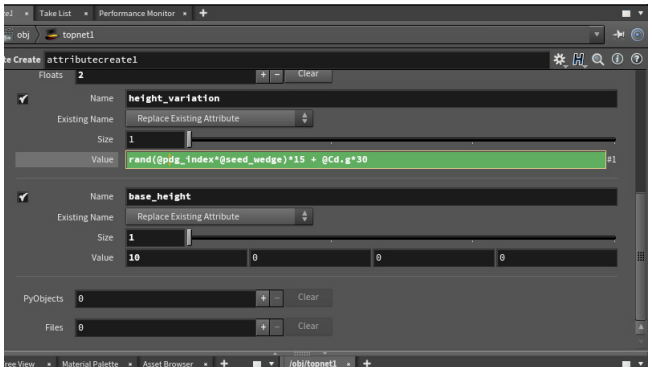
04

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。



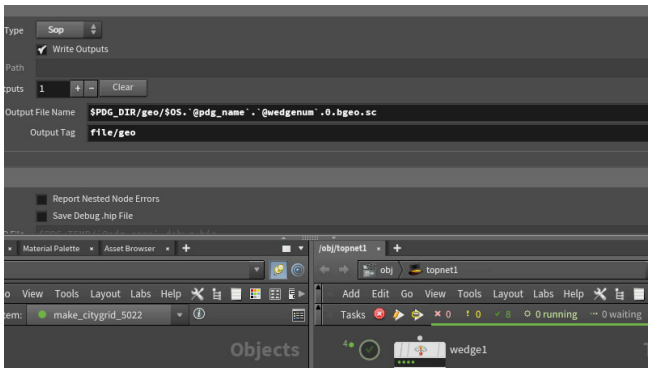
05

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。

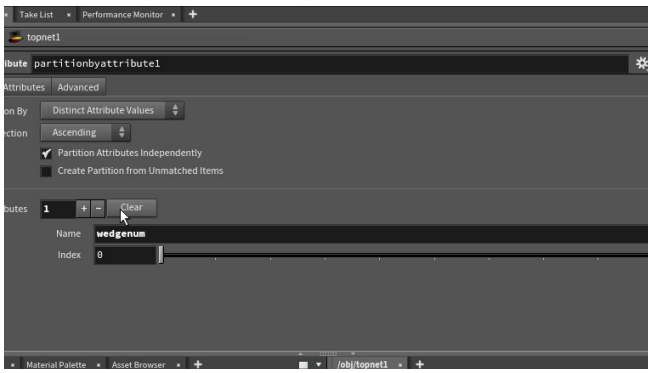
```
rand(@pdg_index*@seed_wedge) * 15 + @Cd.g*30
```



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。

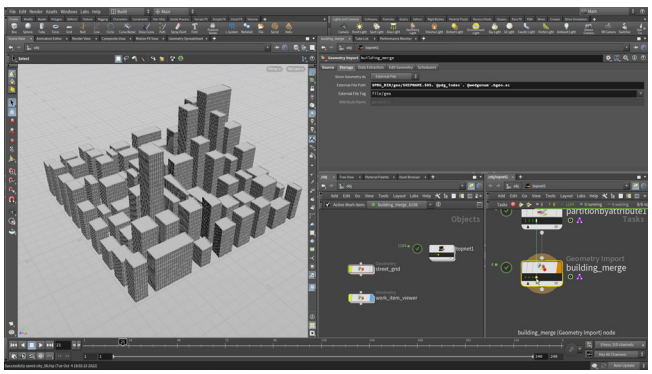
- ***make_citygrid*** HDA Processor
- ***geometryimport*** Geometry Import
- ***create_buildings*** ROP Geometry
- ***building_merge*** Geometry Import



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

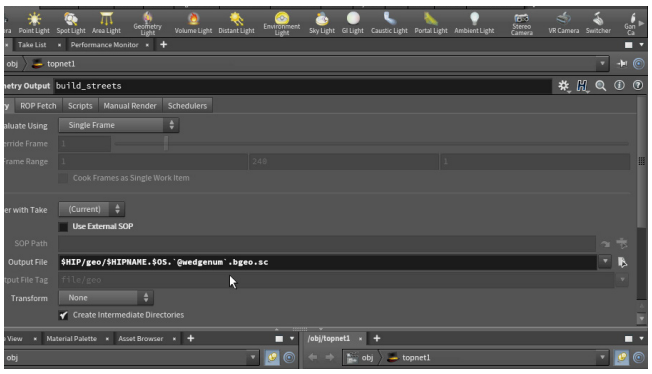
作業内容を**保存**してから、***building_merge*** TOP ノードをクックします。都市マップが 4 つ作成され、***building_merge*** ノードのワークアイテムをクリックすることで視覚化できます。

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

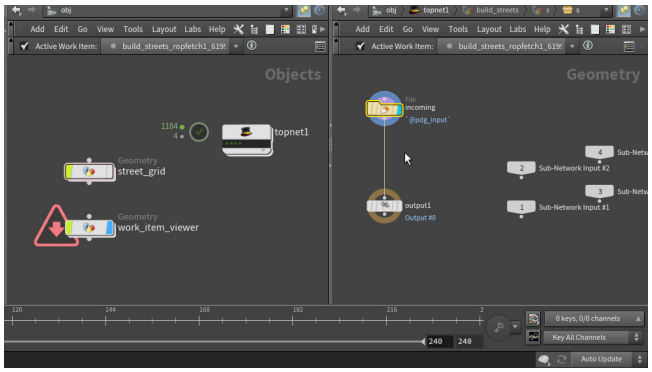
ビルのコンテキストを作成するため、最終レンダリングで使用する都市ブロックと街路を作成します。4 つの Wedge には、現時点ではすべて同じマップが使用されています。このジオメトリを TOP でセットアップして、後で異なるマップを使用できるようにします。非常に堅牢なシステムを作るには、柔軟性を組み込むことが大切です。



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。

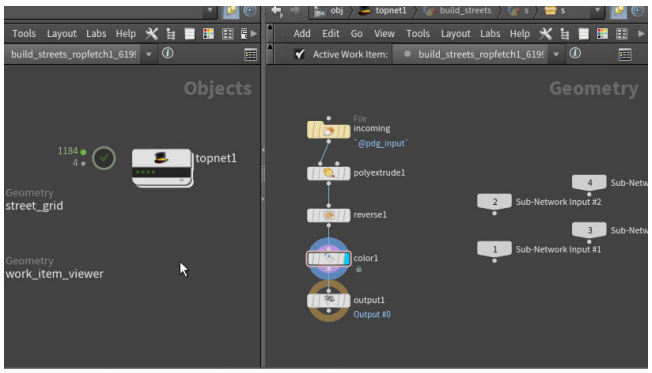
Use External SOP オプションを**オフ**にし、**Output File** のエクスプレッションで、.bgeo の直前に $F ではなく `@wedgenum` を追加します。

ROP Fetch タブで、**Cache Mode** を **Write Files** に設定します。



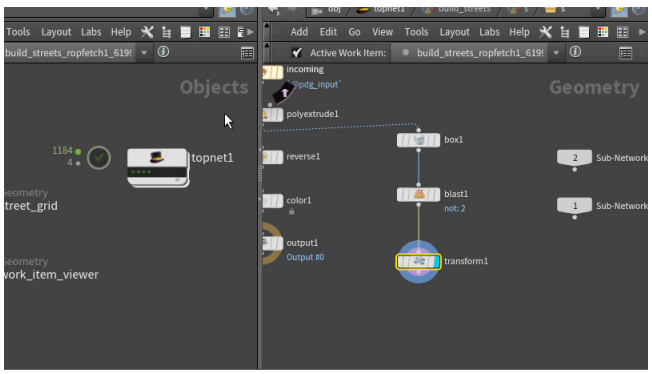
02 このノードを **RMB クリック**して **Generate Node** を選択し、4 つのワークアイテムを作成します。いずれかをクリックしてワークアイテムをハイライトしたら、ノードをダブルクリックしてその中に入ります。

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。

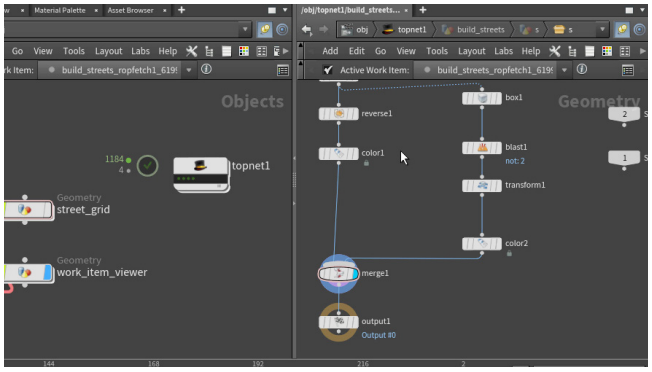


04 ネットワークの端の方に **Box** ノードを追加します。box ノードに ***polyextrude*** を接続し、境界ボックスを都市グリッドのジオメトリに合わせます。

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。

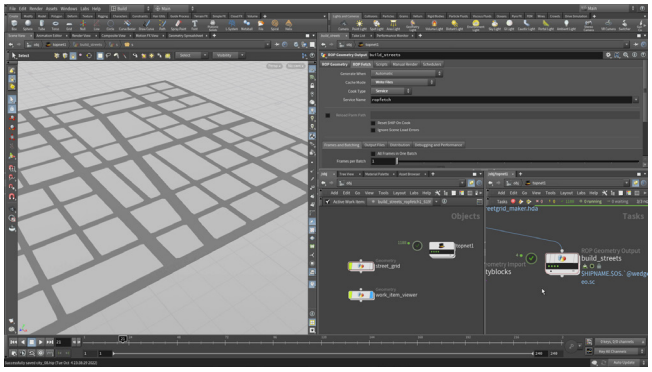
- **Translate Y** を **-0.05** にする
- **Scale X** を **1.05** にする
- **Scale Z** を **1.05** にする



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。

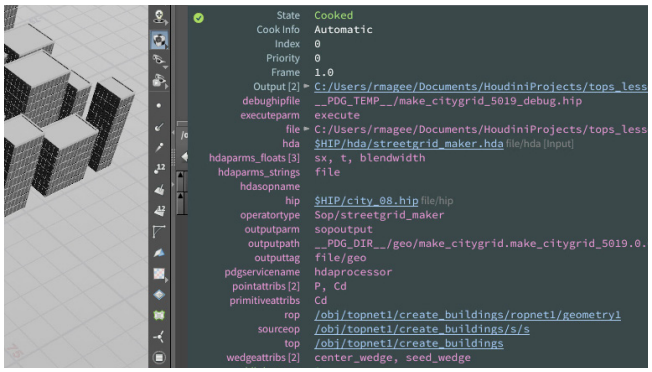
Merge ノードを作成し、2 つの ***color*** ノードに接続します。街路のジオメトリが表示されるよう、**Display フラグ**が出力に設定されていることを確認します。



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。

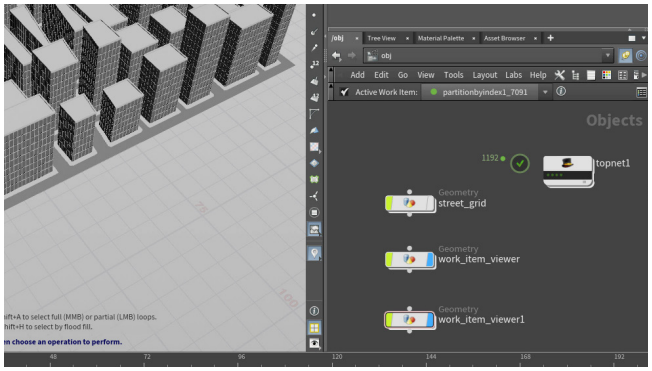
現時点では、これら 4 つはすべて同じです。方程式に都市グリッドを追加すれば、変わっていくはずです。



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。

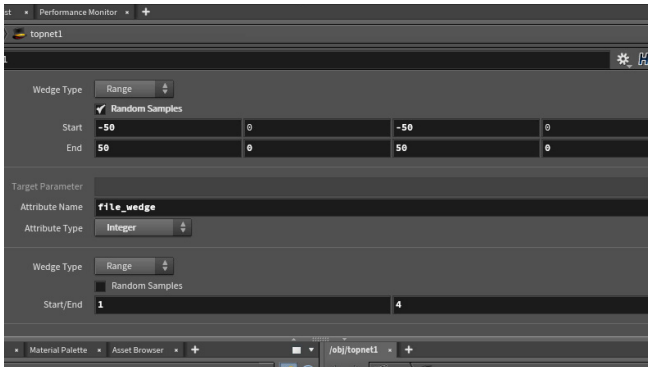
Output [2]

C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/city_01.building_merge.1.1.bgeo.sc file/geo
C:/Users/rob/Documents/HoudiniProjects/tops_lesson/geo/city_01.build_streets.1.bgeo.sc file/geo

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。

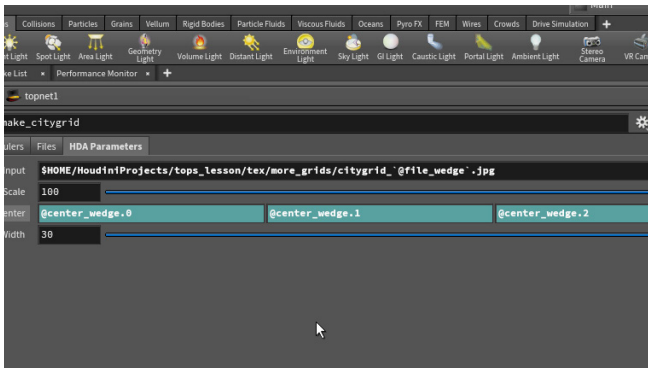


01 ***wedge*** TOP ノードに戻ります。**Wedge Attributes** の横にある **+(プラス)記号**をクリックし、新しい Wedge パラメータを追加します。

次のように設定します。

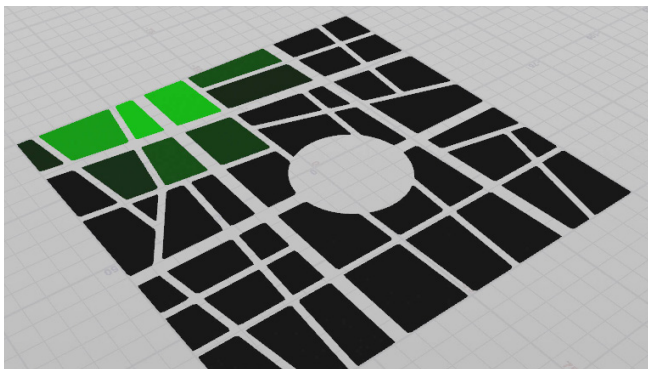
- **Attribute Name** を ***file_wedge*** にする
- **Attribute Type** を **Integer** にする
- **Start/End** を **1, 4** にする

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。

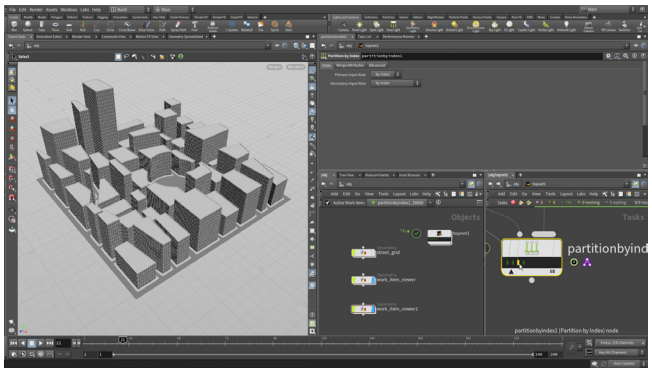


02 ***streetgrid_maker*** HDA Processor ノードで、**Image Input** の横にある**ファイル選択**ボタンをクリックし、***more_grids*** フォルダに移動します。そこからファイルを選択したら、パラメータエディタで、エクスプレッションの `$F` を `` `@file_wedge` `` に変更します。

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。

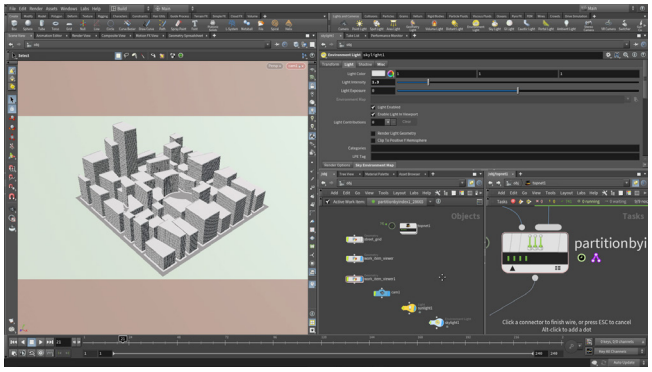


04 2 つ目の ***attributecreate*** TOP ノードを**バイパス**します。これは、静的なマップ向けに設計されたもので、4 つの異なるマップでは正しく動作しません。

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

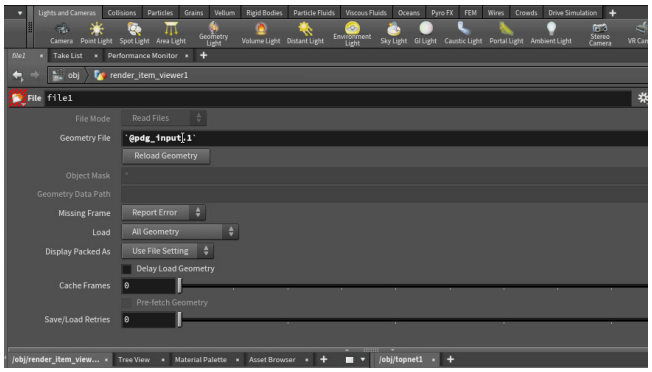
カメラとスカイライトを追加してから、都市をレンダリングします。その結果を単一のモザイクへとまとめますが、Wedge アトリビュートを表示したままにすることで、結果をもとにクリエイティブな決定を下せるようにします。モザイクノードは Image Magick を使用するので、この手順を実行するにはコンピュータにインストールしておく必要があります。



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

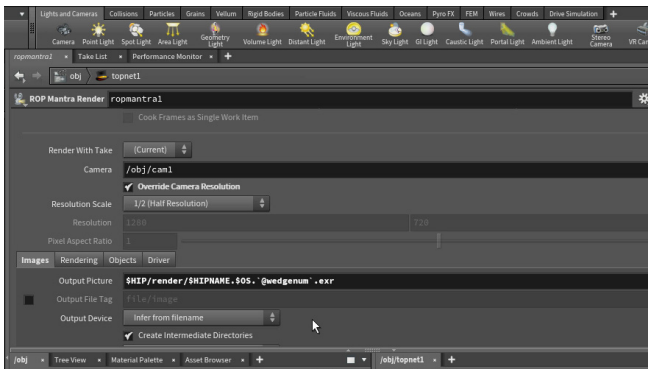
オブジェクトレベルで、***work_item_viewer*** を **Alt ドラッグ**してコピーを作成します。新しいノードを ***render_item_viewer*** という**名前に変更**します。このノードの中に入り、**Geometry File** パラメータを次のように変更します。

```
`@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。

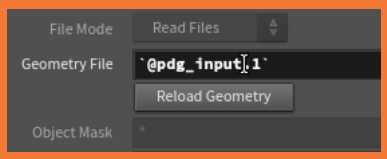
```
$HIP/render/city.$OS.`@wedgenum`.exr
```

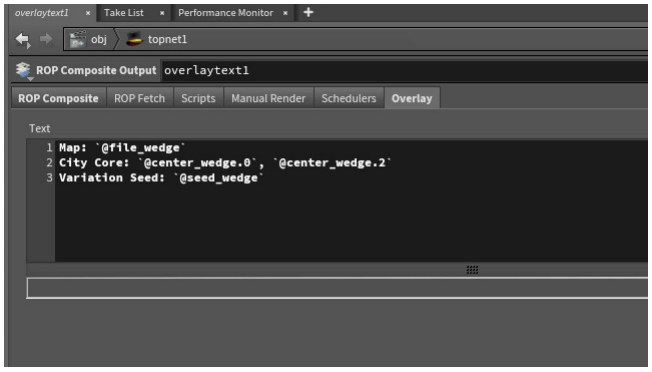
注: Houdini Apprentice を使用してこのチュートリアルを実行している場合は、`.exr` ではなく `.pic` ファイルを使用して、Apprentice のウォーターマークが追加されないようにしてください。

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

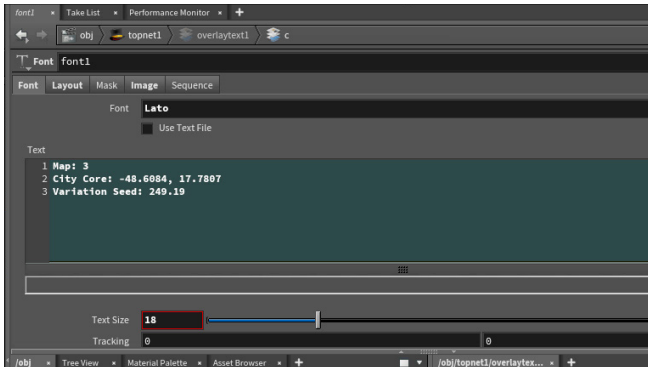
Overlay Text Top を追加し、**Output Picture** を次のように設定します。

```
$HIP/render/$HIPNAME.$OS.`@wedgenum`.exr
```

ROP Fetch タブで、**Cache Mode** を **Write Files** に設定します。

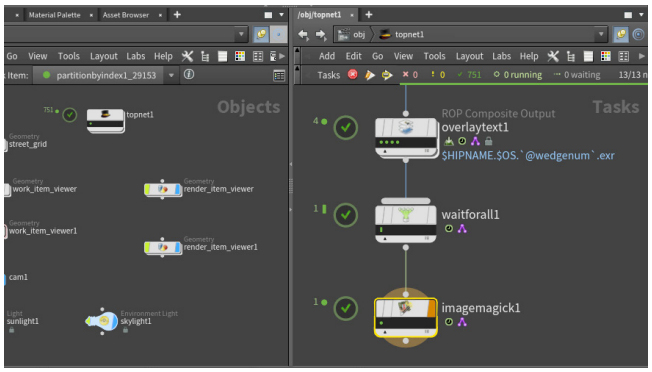
Overlay タブで、次のように入力します。

```
Map: `@file_wedge`
City Core: `@center_wedge.0`, `@center_wedge.2`
Variation Seed: `@seed_wedge`
```



05

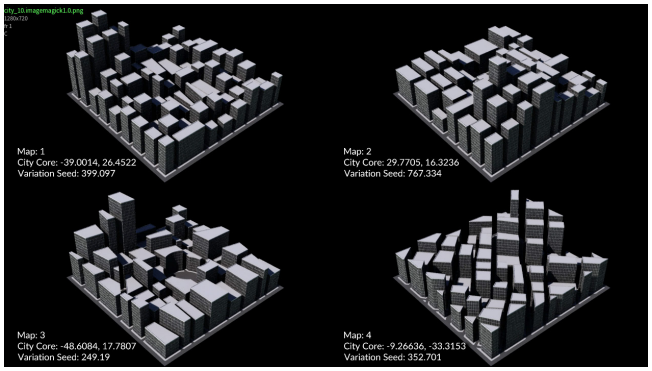
Overlaytext ノードを**ダブルクリック**して、中に入ります。***Font*** ノードを選択し、**Text Size** を **18** に変更します。これでテキストが小さくなり、都市の表示スペースが広くなります。



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

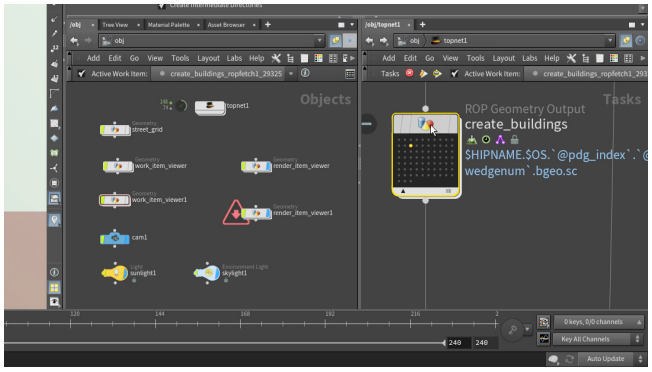
PDG は Python に対応しているため、Houdini と直接接続していないタスクにも使用できます。ワークフローを整理する TOP ノードによって、パイプラインツールを整理するための視覚ツールを提供します。

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

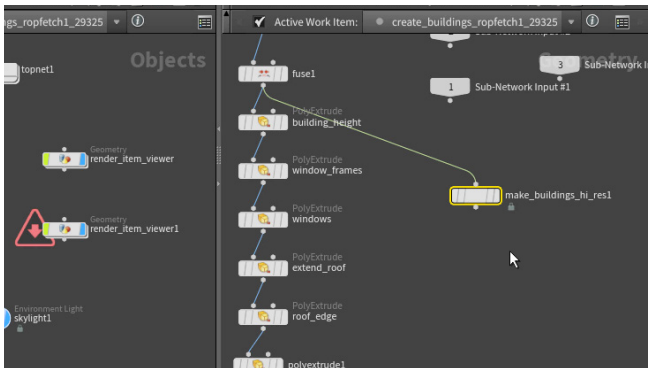
スケールアップでより多くのコンテンツを作成

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。



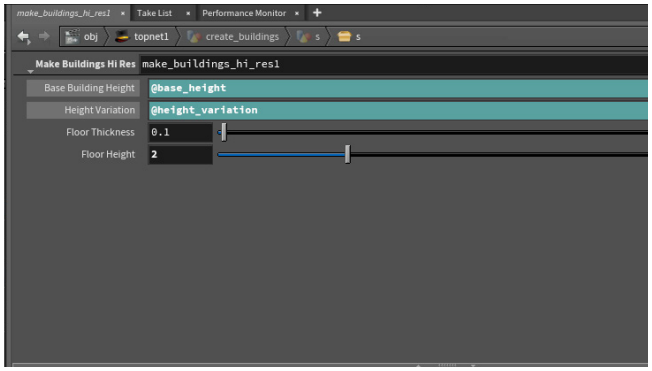
01 Wedge TOP で、Wedge Count を 1 に設定します。これにより、新しいマップを使用して都市をスケールアップする間、反復回数が 1 回のみになります。

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。

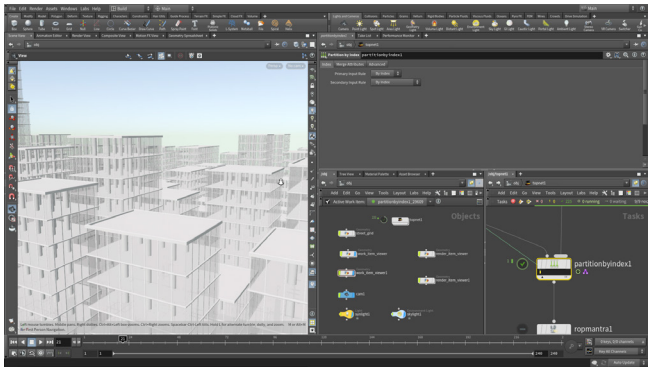
fuse ノードを新しいアセットに接続します。



03 ***make_buildings_hi_res*** デジタルアセットで、次のように設定します。

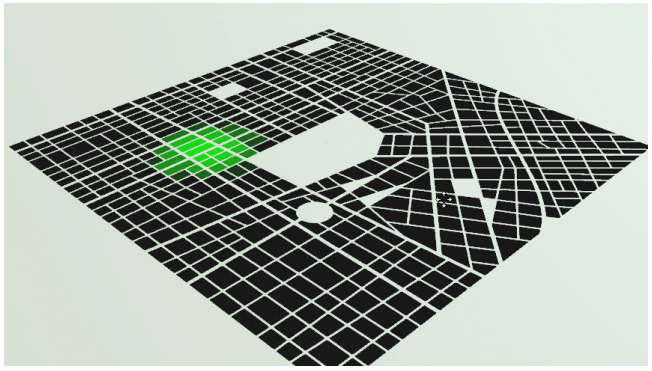
- **Floor Height** を **2** にする
- **Base Building Height** を `@base_height` にする
- **Height Variation** を `@height_variation` にする

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。

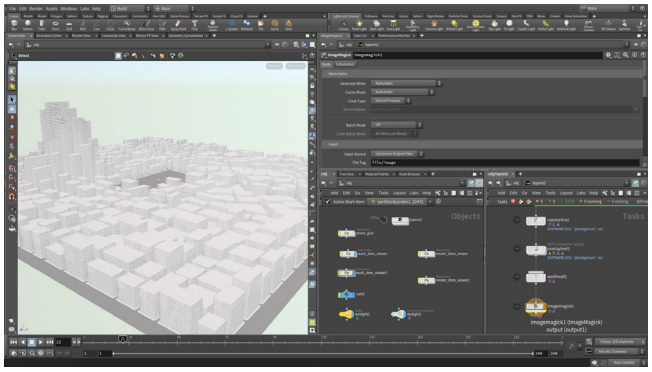
多くのジオメトリが生成されました。演算ファームを使用してワークアイテムの数を増やす利点がわかり始めたのではないでしょうか。他のスケジューラ系ノードの 1 つを使って、それを実現できます。



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。

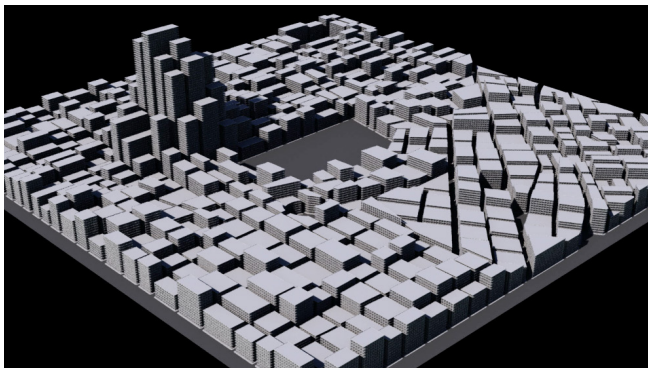
Wedge ノードで、***center_wedge*** アトリビュートを -110, 0, 100 などに設定することもできます。



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。