



UNREAL
ENGINE

Unreal Engine 5.2 アップデート ～Rendering/PCG～

Epic Games Japan

Electric Dreams - 環境サンプル

ニュース

サンプル

マーケットプレイス

ライブラリ

• Twinmotion

起動
Unreal Engine 5.2.0



UE 機能サンプル



Electric Dreams - UE5.2

- **Substrate**

- ・新リアルオーサリングシステム

- **PCG**

- ・プロシージャルにアセットを配置



Substrate - 新しいマテリアルオーサリングシステム

実験的機能

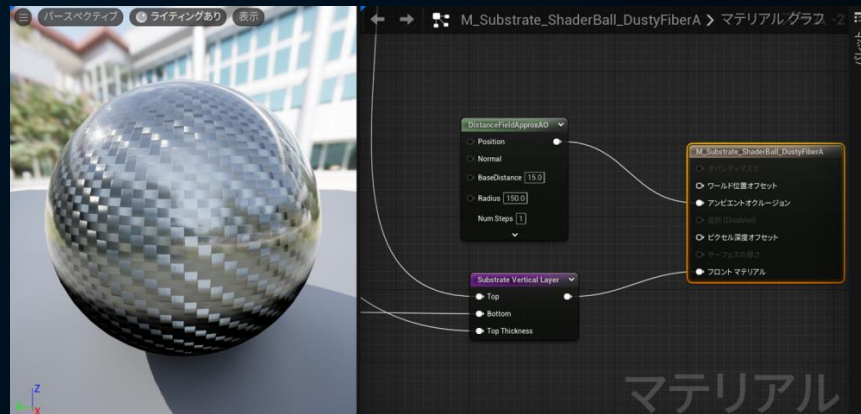


- Substrate Slab BSDF
- Diffuse Albedo
- F0
- F90
- Roughness
- Anisotropy
- Normal
- Tangent
- SSS MFP
- SSS MFP Scale
- SSS Phase Anisotropy
- Emissive Color
- Second Roughness
- Second Roughness Weight
- Fuzz Roughness
- Fuzz Amount
- Fuzz Color

- M_BlendMossFuzz
- オパティマスク
- ワールド位置オフセット
- アンビエントオクルージョン
- 屈折 (Disabled)
- ピクセル深度オフセット
- サーフェスの厚さ
- フロント マテリアル

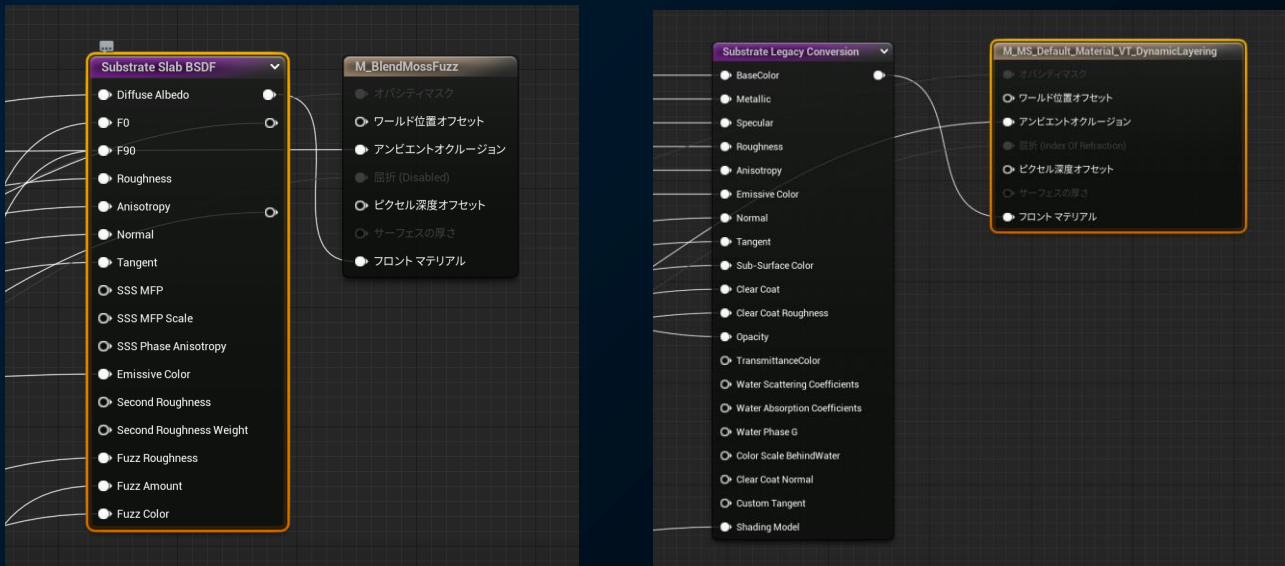
Substrate - 目的

- ・ 固定化されていたシェーディングモデルをモジュール式に置き換え
- ・ 表現力の向上を目指す
→より複雑な混合、レイヤード表現をリアルタイムで
- ・ 互換性とパフォーマンス
→従来と同様の使い方であれば
同程度のコストで表現できるように



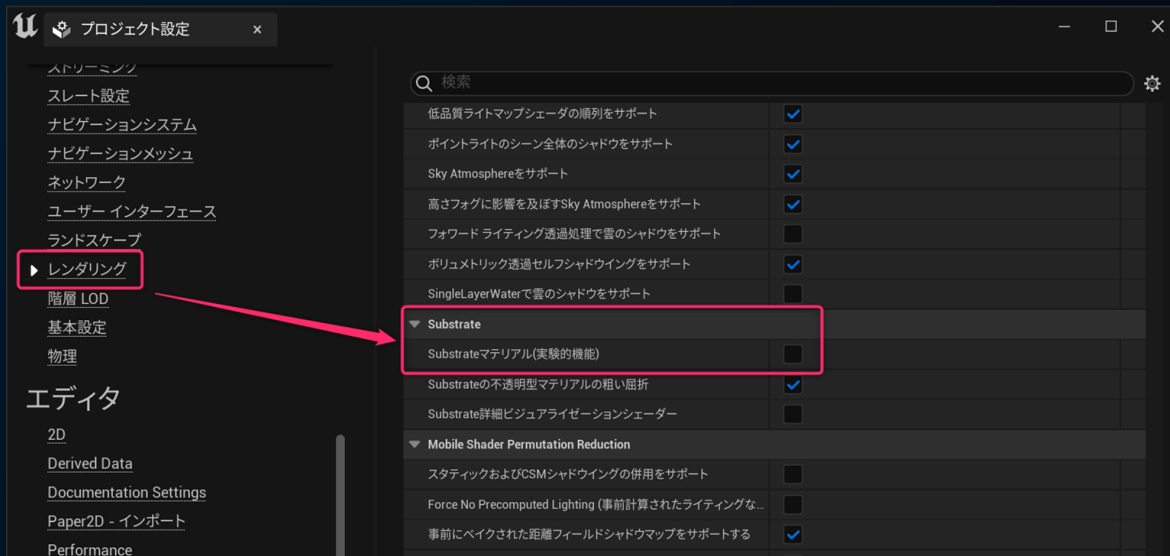
Substrate - Electric Dreams 環境サンプル

- ・ Electric Dreamでも使用されているが...
従来の材料から変換した材料も多く存在
→ 複雑なものが多いため、初めて見る場合は**機能別サンプル**をおすすめ



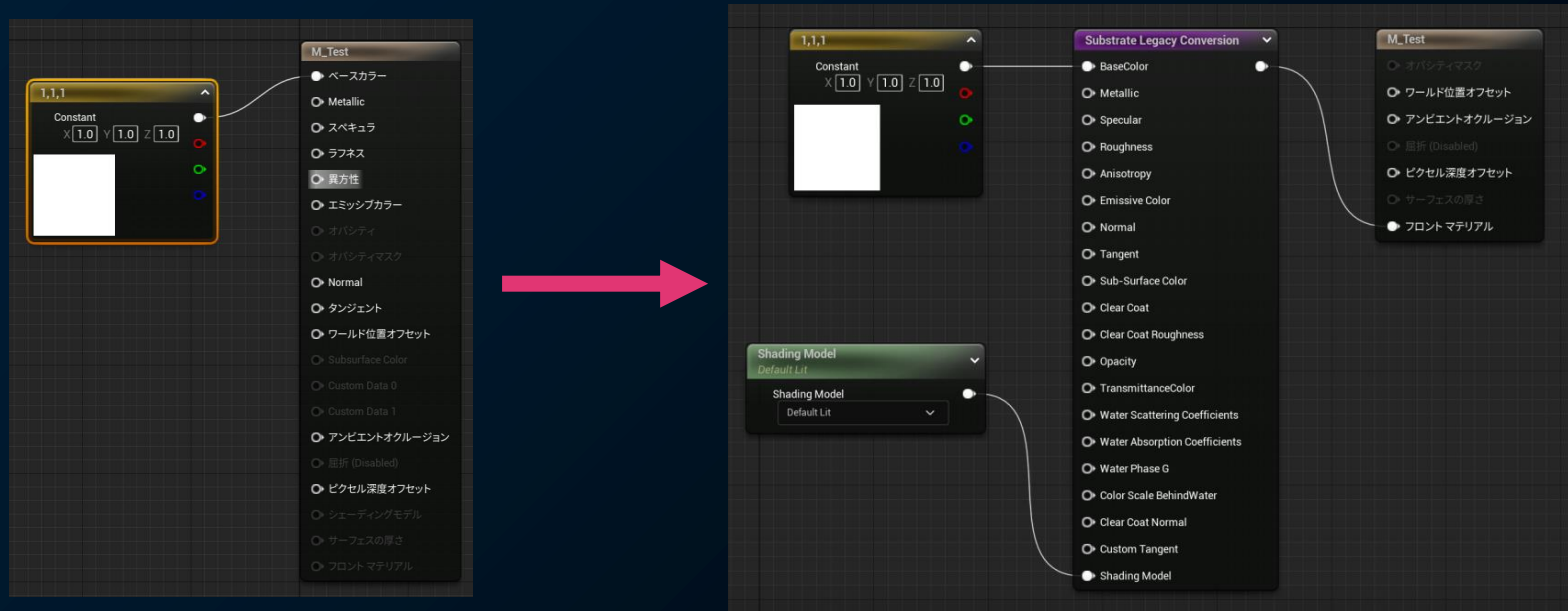
Substrate - 機能別サンプル

- 機能別サンプルプロジェクトの **/Maps/SubstrateMaterials** マップ
デフォルトでは機能が有効化されていないため
プロジェクト設定から設定を有効化した後、マップを開く



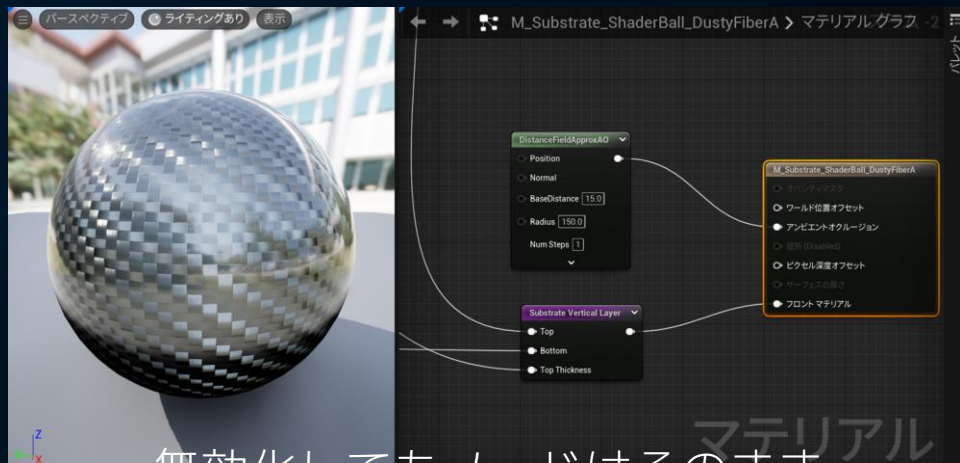
Substrate - 注意点

- ・ 実験的な機能のため、検証での利用を想定
既存のマテリアルもSubstrate形式に変換される



Substrate - 注意点

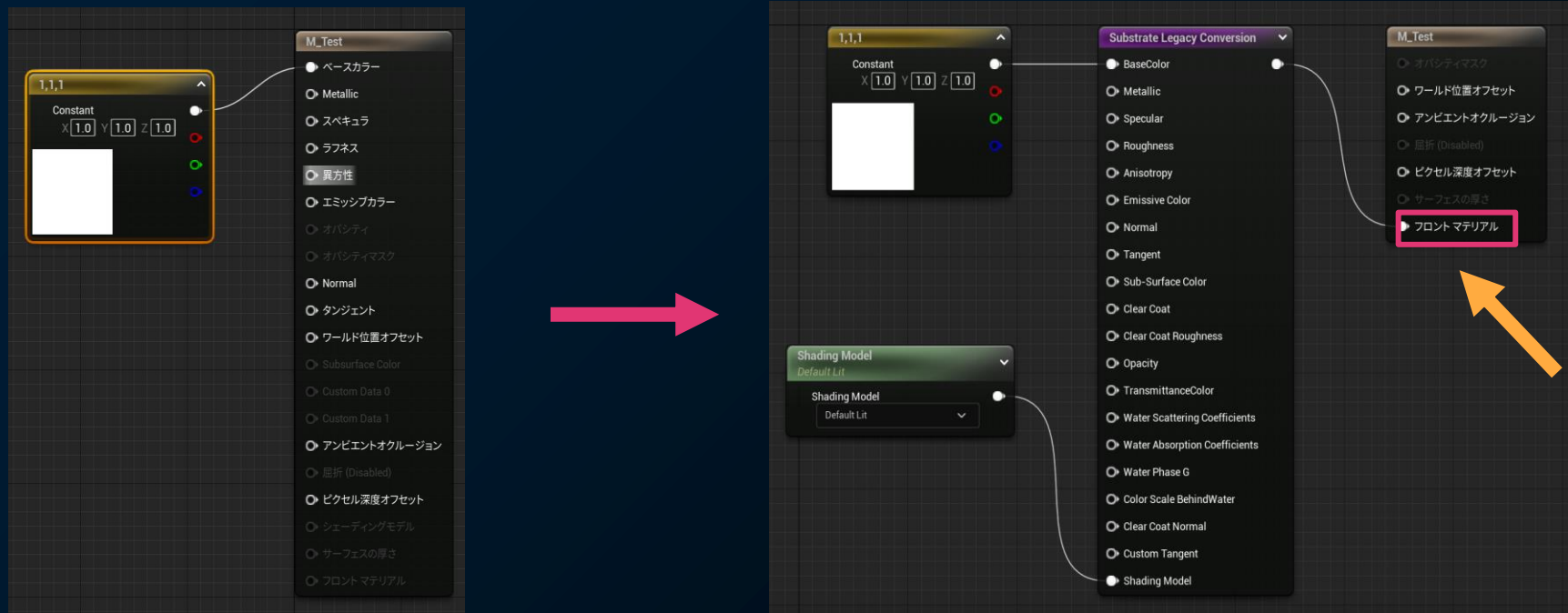
- 変換した既存マテリアル保存してしまうと、Substrate機能をプロジェクトから無効化しても**自動でマテリアルは戻らない**



無効化してもノードはそのまま

Substrate - 変換されたマテリアル

- Substrateマテリアルでは最終出力ノードを**フロントマテリアル**に設定



Substrate - Substrate BSDF ノード

- これまでのマテリアルで設定していたシェーディングモデルはノードで指定 **Substrate BSDF**(双方向散乱分布関数)ノードとして用意されている

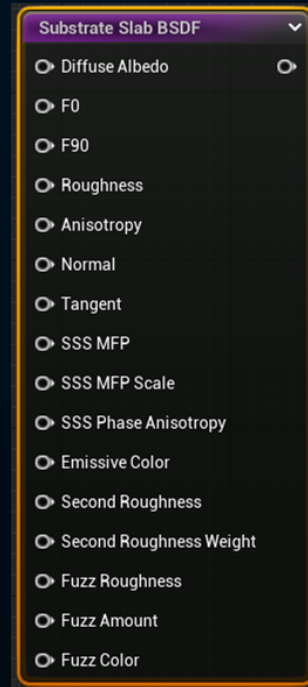
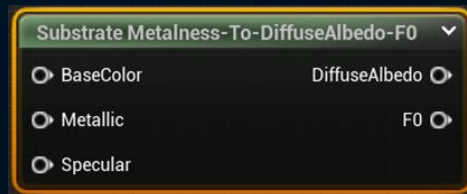


Substrate - Slab

- ・ デフォルトのBSDFノードは、**Slab**

Metallicのようなパラメータがなくなり、
これまでのマテリアルとは異なるパラメータ化が行われている

ヘルパーノードで従来の指定方法もエミュレート可能

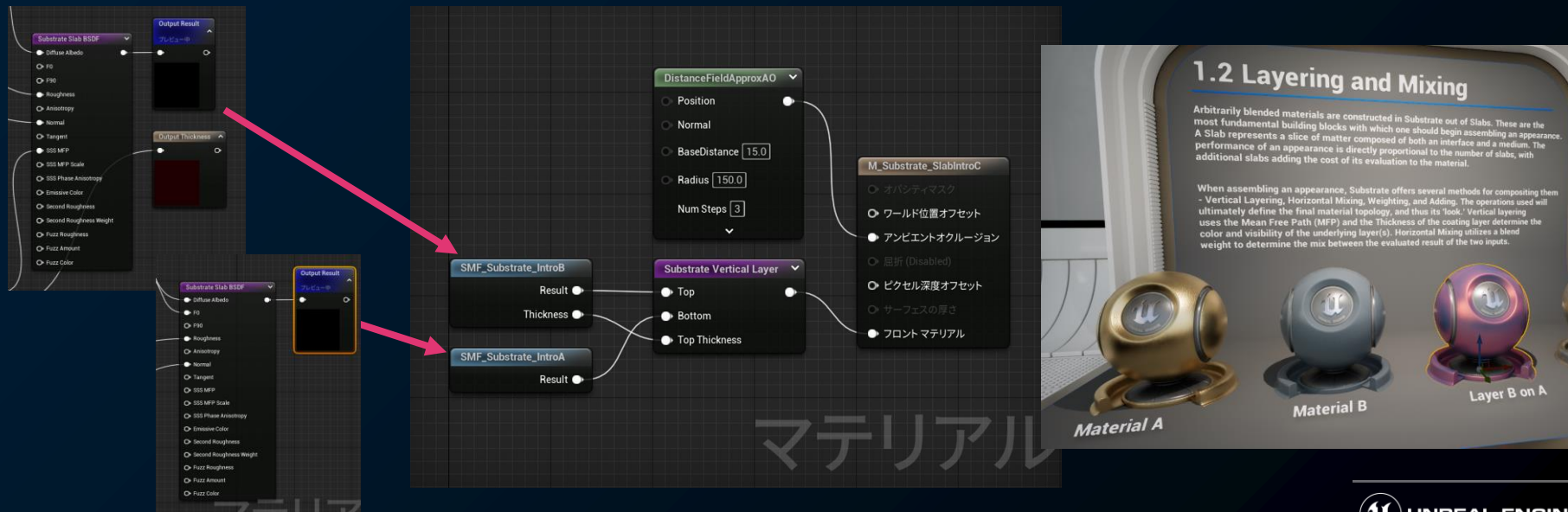


Substrate - Material Nodes - Substrate Materials

<https://dev.epicgames.com/community/learning/courses/92D/unreal-engine-substrate-materials/8rXW/unreal-engine-substrate-the-slab>

Substrate - Substrate Vertical Layer

- 機能別サンプルの例 - Layer B on A マテリアル
Slub B SDF ノードを出力する Material Function を作成しておき、
それぞれを重ね合わせる **Substrate Vertical Layer** ノードに繋いでいる



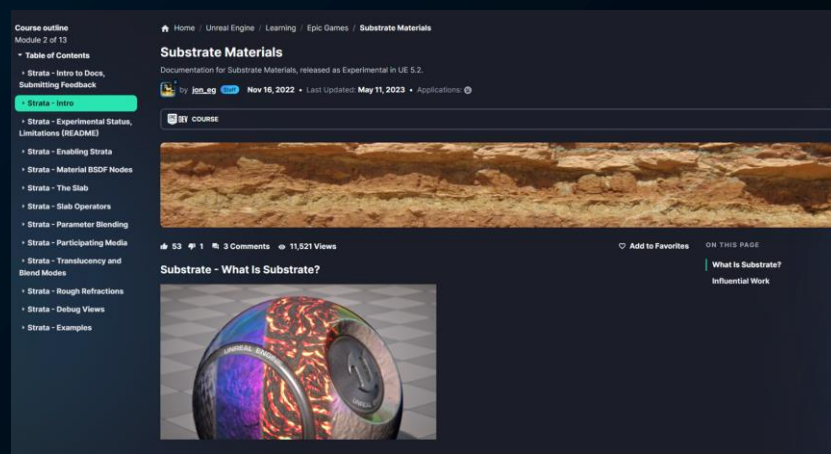
Substrate - ドキュメントとFeedback Thread

- ・現時点のドキュメントはEDCにて公開

<https://dev.epicgames.com/community/learning/courses/92D/unreal-engine-substrate-materials/vEaW/unreal-engine-substrate-what-is-substrate>

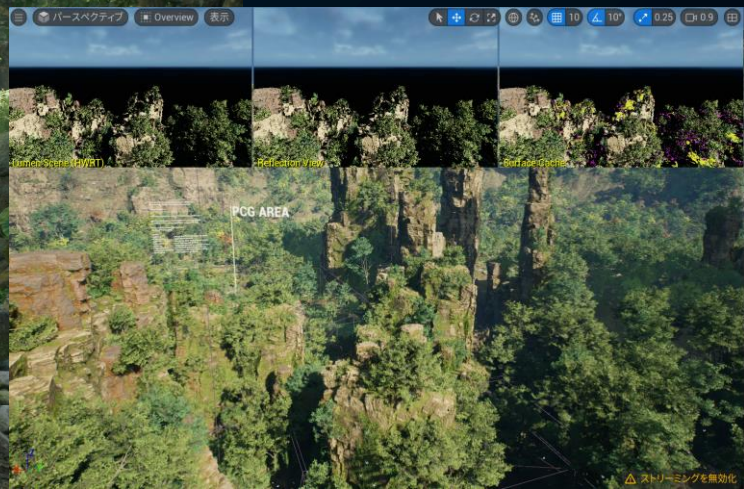
- ・開発者によるFeedback Threadにて様々な質問や返答が行われている

<https://forums.unrealengine.com/t/substrate-feedback-thread/691314>



The image shows a screenshot of the Unreal Engine Substrate Materials documentation page and a forum thread. The documentation page on the left lists a course outline with 13 modules, with the first module 'Substrate - Intro' highlighted. The forum thread on the right is titled 'Substrate - What is Substrate?' and features a large image of a textured surface and a smaller image of a sphere with a glowing, colorful interior. The forum post has 53 likes, 1 reply, and 3 comments, with 11,521 views. The Unreal Engine logo is visible in the bottom right corner.

Lumen



Lumen - アップデート内容

- ・ GIやオクルージョンの品質向上
- ・ 高品質透過性反射のラフネス対応
- ・ 反射の品質向上(HWRT)
- ・ Two Sided Foliage対応(HWRT)

Lumen - GIやオクルージョンの品質向上

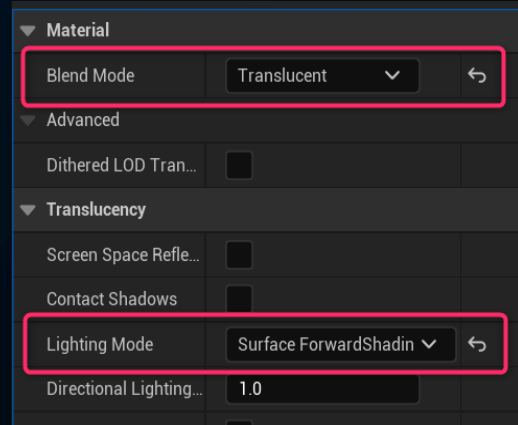
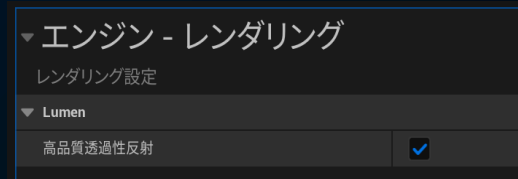
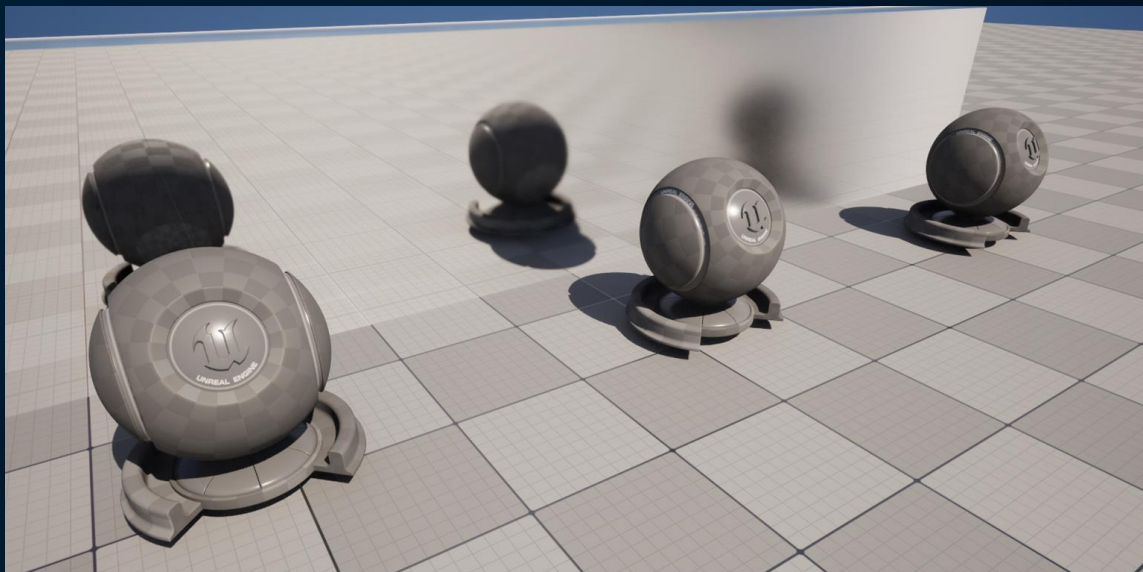
- ・ 薄いジオメトリに対する GI とオクルージョンが向上し、Hair Groomとの統合も改善



5.2リリースノートより

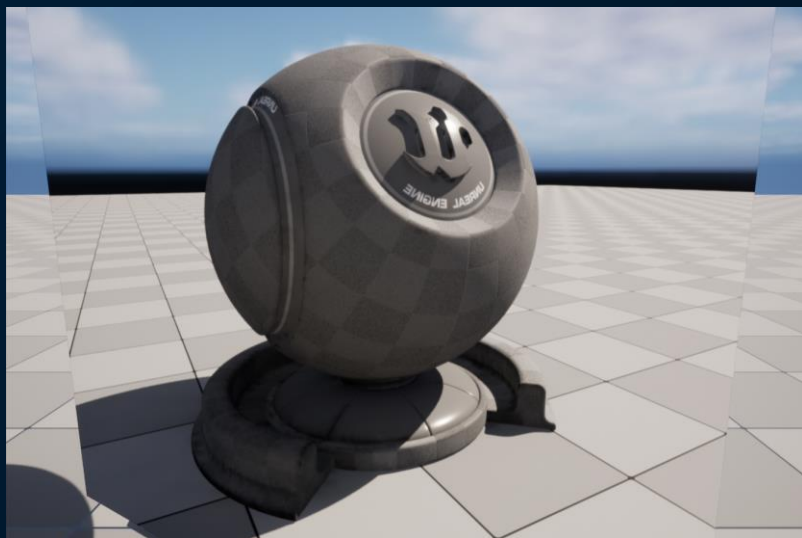
Lumen - 高品質透過性反射のラフネス対応

- ・ 高品質透過性反射がラフネスに対応
(High Quality Translucency Reflections)

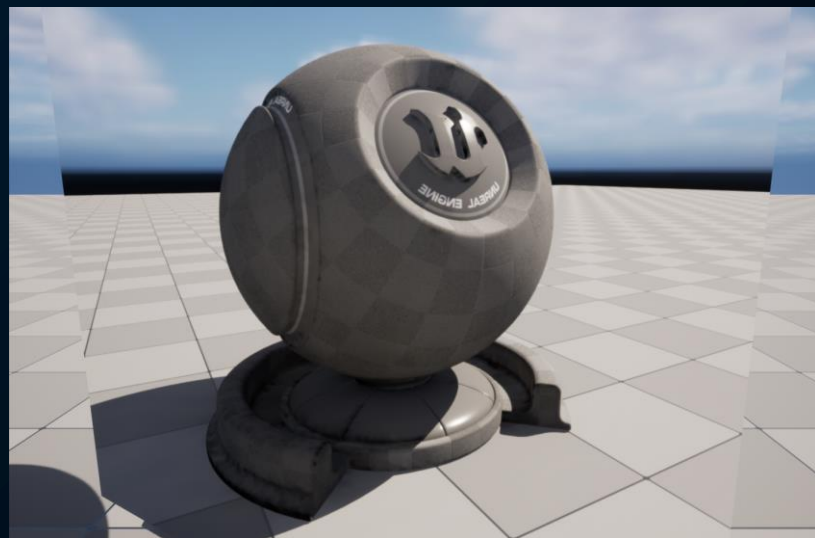


Lumen - リフレクションの品質向上(HWRT)

- ・ハードウェアレイトレーシング (HWRT) ヒットライティング設定時
リフレクション(反射)に含まれるセカンダリバウンスの近似処理で品質を向上



UE5.1



UE5.2

Lumen - リフレクションの品質向上(HWRT)



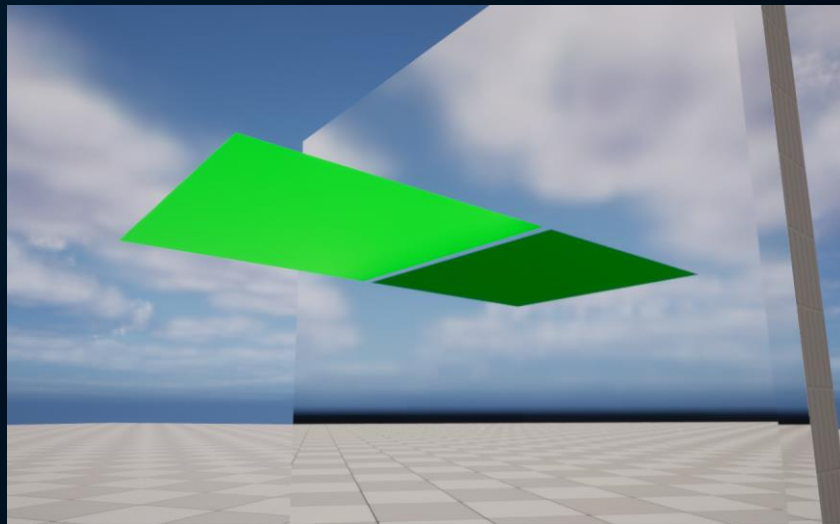
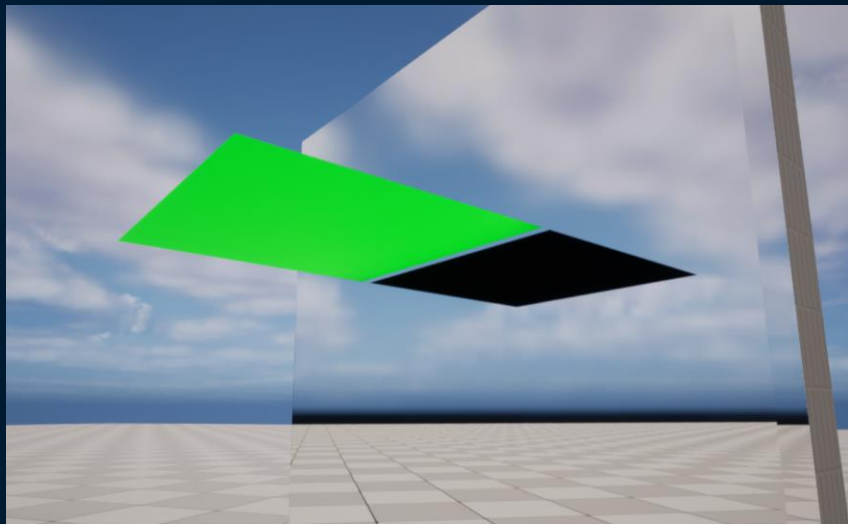
UE5.1



UE5.2

Lumen - Two Sided Foliage対応(HWRT)

HWRT-ヒットライティング設定時、**Two Sided Foliage**が反射に対応



Lumen - パフォーマンスガイドの紹介

パフォーマンスを上げたい場合はこちらをチェック



UNREAL ENGINE 5

Lumen パフォーマンスガイド

Lumen グローバルレイルミネーションと反射に関するスケーラビリティ オプションを説明します。

Lumen では、グローバルレイルミネーション、不透明型マテリアルおよび透過マテリアル、ボリュメトリックフォグに対して、1080p の 8ms および 4ms のフレームバジェットで、コンソールで 30 fps (フレーム/秒) および 60 fps をターゲットとしています。エンジンでは、事前定義されたスケーラビリティ設定を使って Lumen のターゲット FPS を制御します。Epic スケーラビリティ レベルでは 30 fps を、High スケーラビリティ レベルでは 60 fps をターゲットにします。

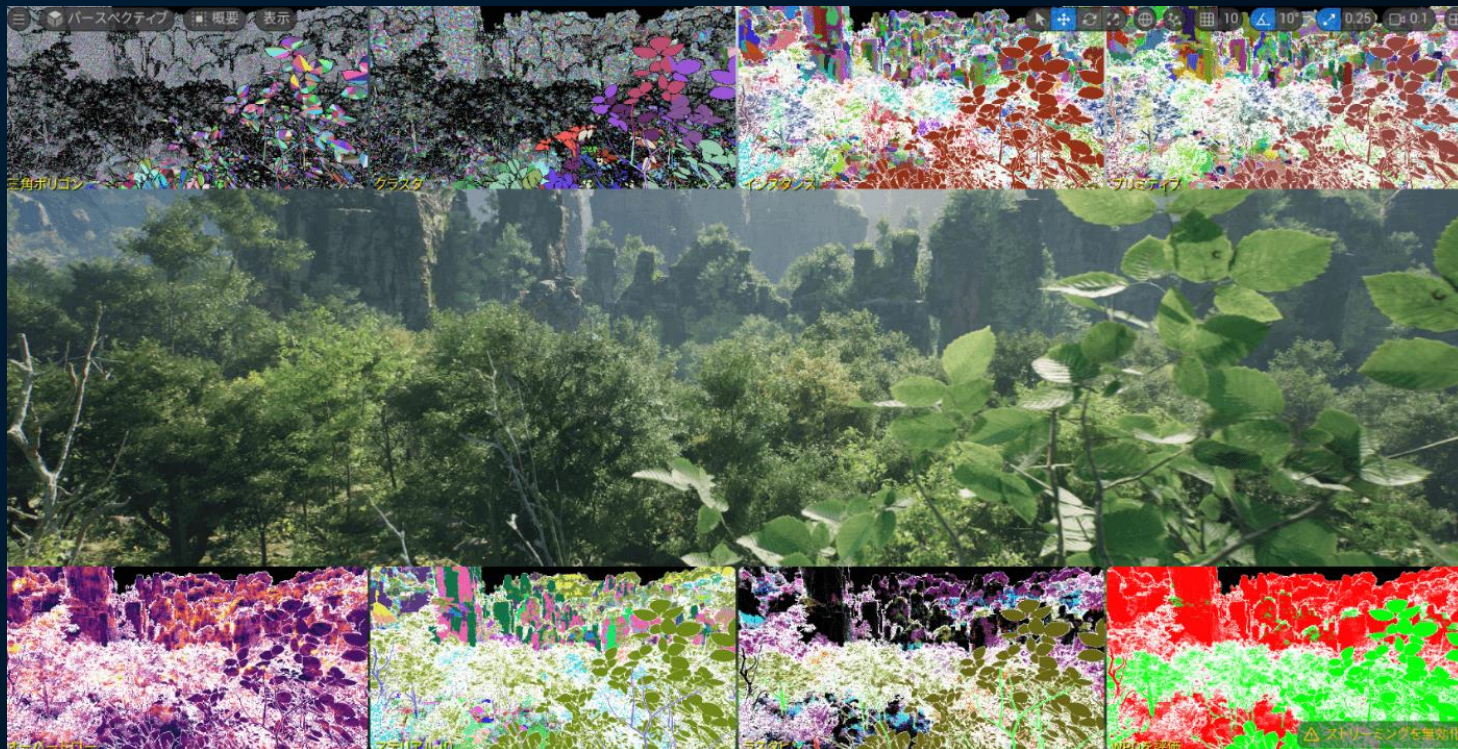
4K 出力については、Lumen は Unreal Engine 5 の **テンポラルスーパー解像度 (TSR)** を使った **テンポラルアップサンプリング** に依存します。Lumen と他の機能ではより低い内部解像度 (1080p) が使用され、これによって TSR で最高の最終画質が実現します。それ以外の場合、これらの機能を 4K でネイティブにレンダリングするには、30 または 60 fps を達成するためにより低い品質設定が必要になります。

コンテンツ

- スケーラビリティ設定
- Lumen よりも低くスケールダウンする
- ソフトウェアレイトレーシング
- ハードウェアレイトレーシング

<https://docs.unrealengine.com/5.2/ja/lumen-performance-guide-for-unreal-engine/>

Nanite

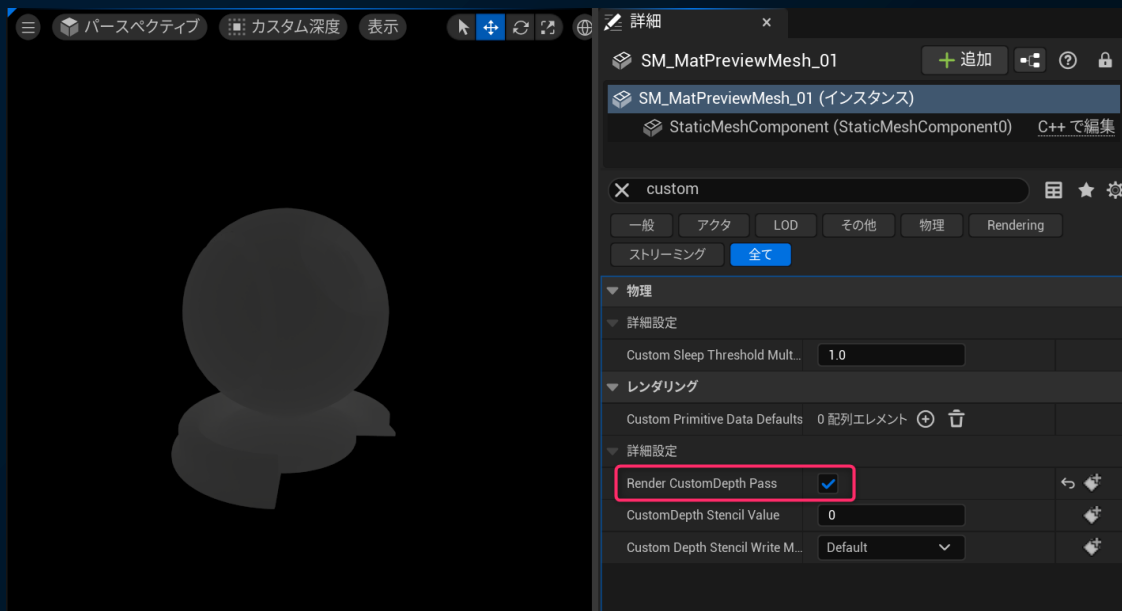


Nanite - アップデート

- ・ Custom DepthとStencil対応
- ・ ライト チャンネル
- ・ Planar Reflectionsのサポート
- ・ 可変精度法線
- ・ Max World Position Offset Displacement
- ・ Stat NaniteStreamer
- ・ Nanite ディスプレイメントマッピング(実験的機能)

Nanite - Custom DepthとStencil対応

- ・通常のStatic Meshと同様に**Custom Depth**や**Stencil**に描画可能に

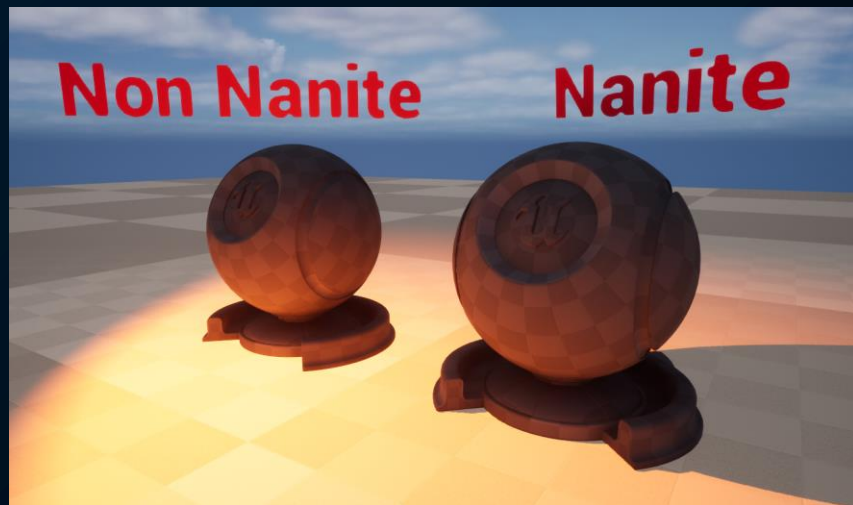


Nanite - ライト チャンネル

- ・ **ライトチャンネル**のNanite対応
どのライトチャンネルに設定してもライティングされてしまっていたのを改善



UE5.1



UE5.2

Nanite - Planar Reflectionsのサポート

- ・ **Planar Reflections**のNanite対応 (非Lumen環境)
5.2でNaniteも反射を描画できるように

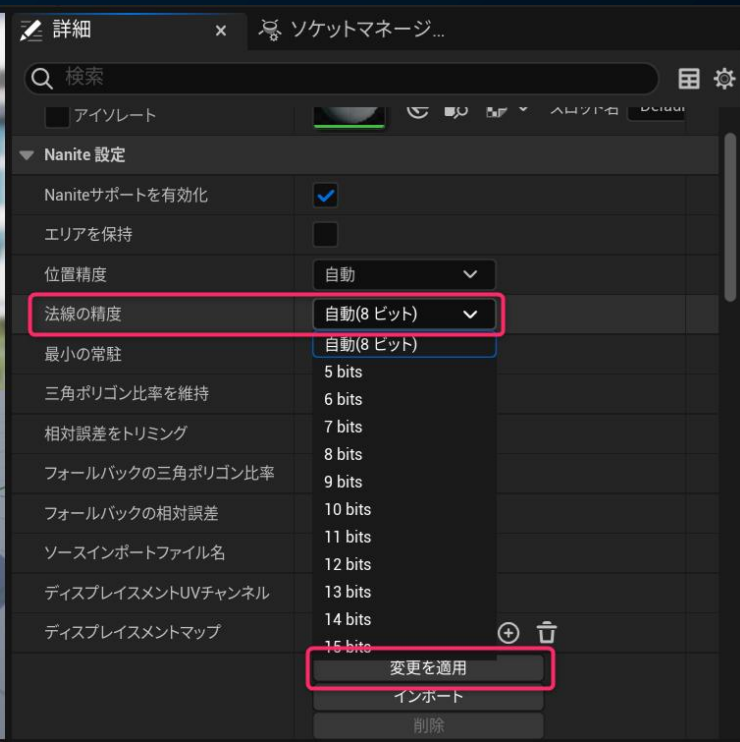
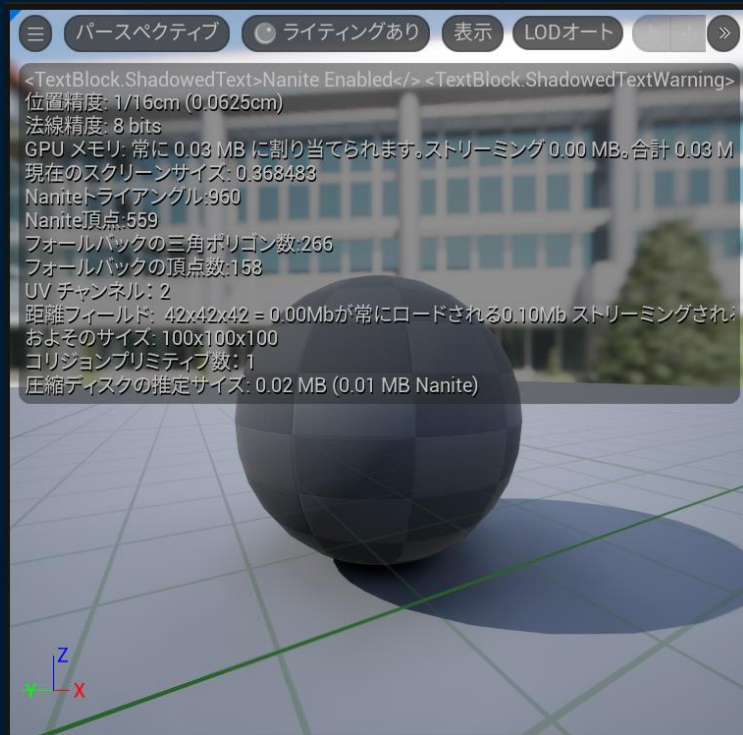


UE5.1

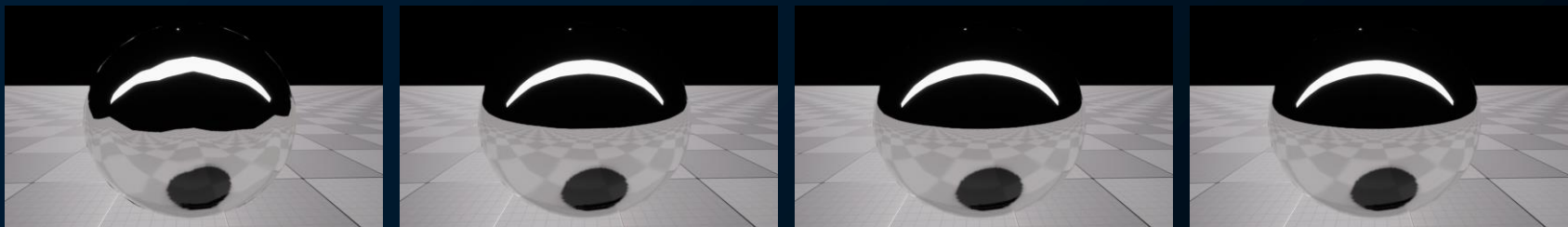


UE5.2

Nanite - 可変精度法線



Nanite - 可変精度法線



5 bits

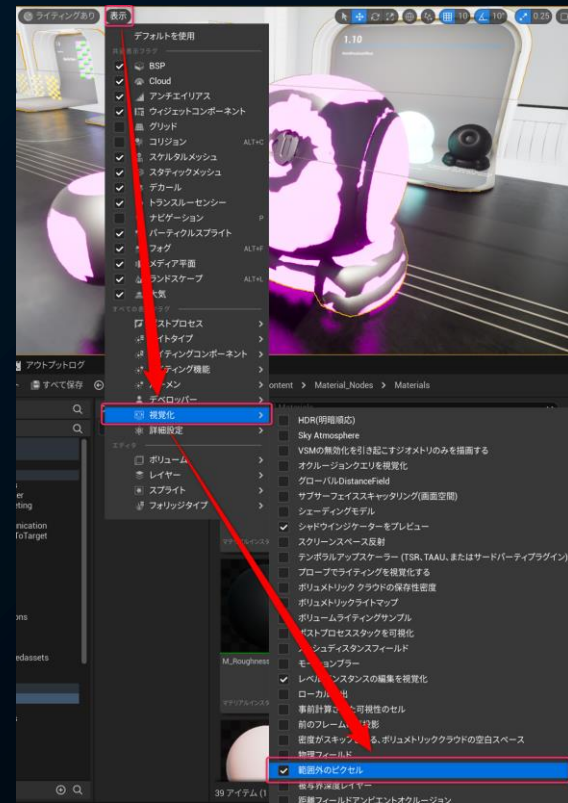
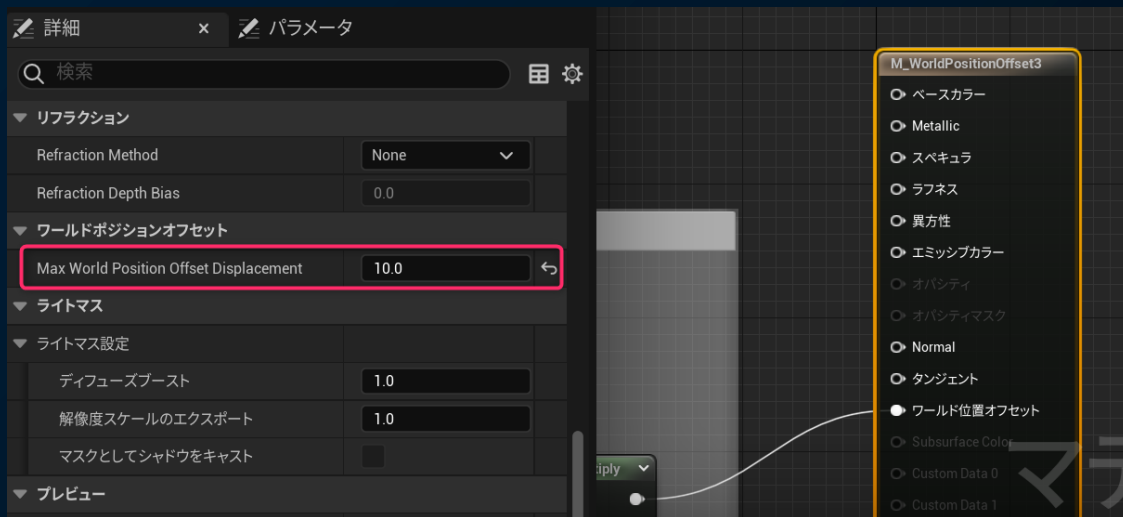
8 bits
デフォルト値

10 bits

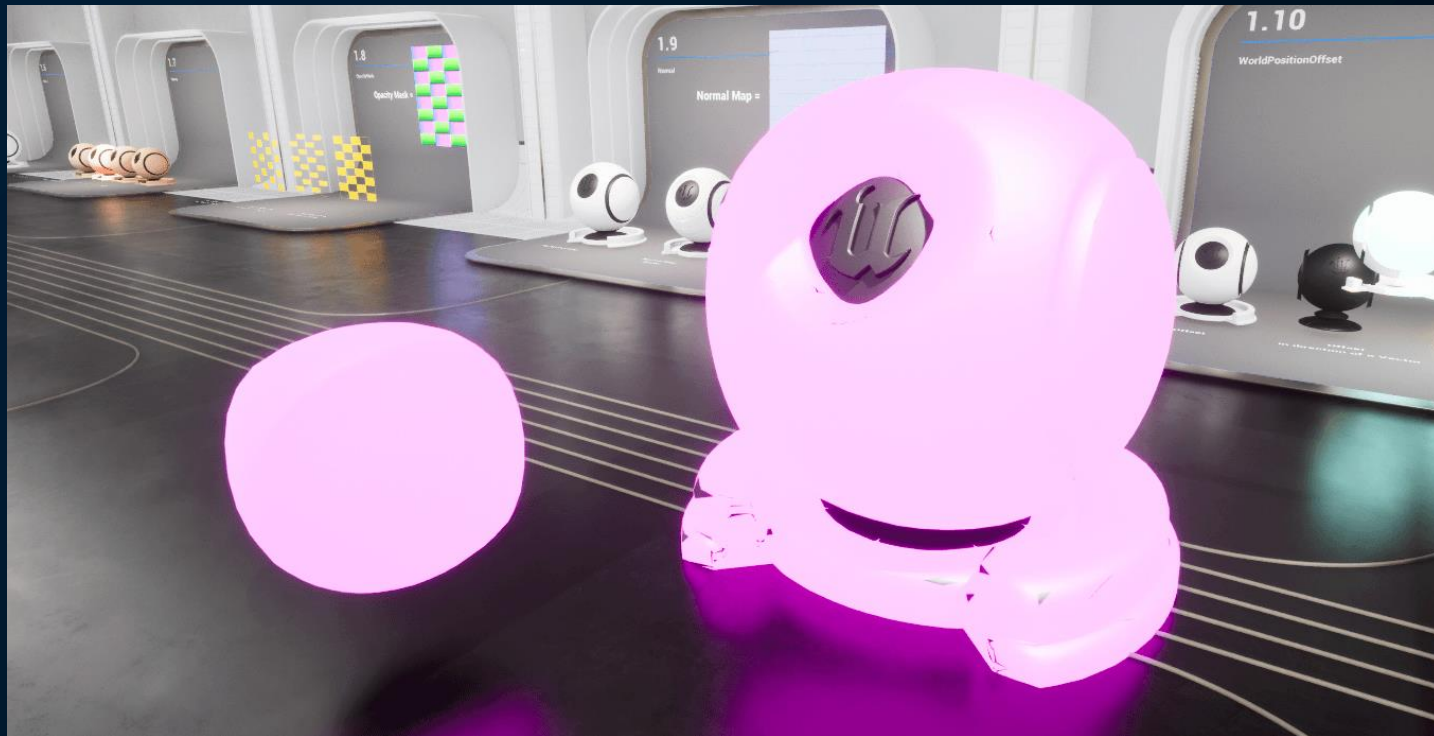
15 bits

Nanite - Max World Position Offset Displacement

- ・ カリング等で問題になるためWPOの移動距離を制限
マテリアル単位で設定
[表示] -> [視覚化] -> [範囲外のピクセル]で可視化



Nanite - [表示] -> [視覚化] -> [範囲外のピクセル]



Nanite - Nanite Streamerの改善

- ・ Nanite Streamerの整備が行われ様々なNanite 情報が見れるように
コマンド ***stat Nanitestreaming***



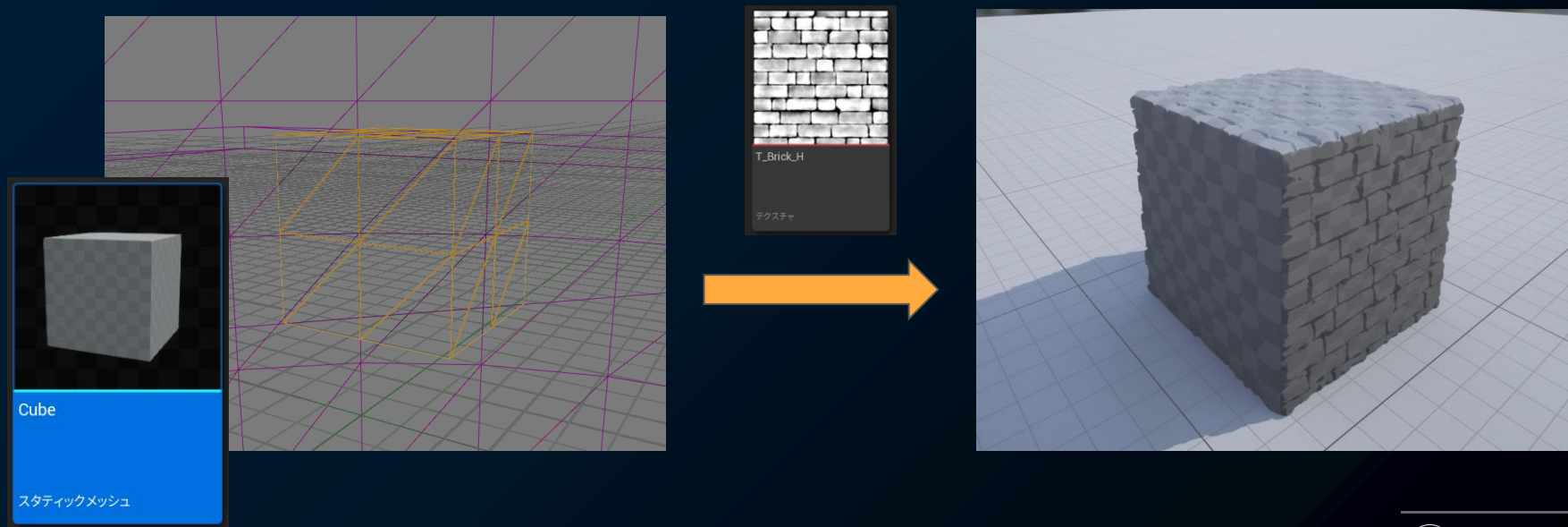
```
NaniteStreaming [STATFP-01F-readstreaming]
Cycle counts/Unit:          Count      InclusiveAvg  InclusiveMax  ExclusiveAvg  ExclusiveMax
AsyncUpdate                1000000    1.58 ms       2.78 ms       0.48 ms       0.79 ms
ProcessRequests            1000000    1.08 ms       2.15 ms       0.55 ms       0.93 ms
AddParentRequests          1000000    0.88 ms       1.22 ms       0.63 ms       1.22 ms
EndAsyncUpdate              1000000    0.02 ms       0.04 ms       0.02 ms       0.04 ms
BeginAsyncUpdate            1000000    0.03 ms       0.04 ms       0.01 ms       0.02 ms

Counters:
Nanite Resources:          842.00     842.00
Imposters                  0.00       0.00
Root Pages:                842.00     842.00
Peak                       842.00     842.00
Allocated                   2,048.00   2,048.00
Streaming Pool Pages       4,096.00   4,096.00
Total Pool Size (MB)       576.00     576.00
Root Pool Size (MB)        64.00      64.00
Streaming Pool Size (MB)   512.00     512.00
Page Requests              117,291.58 160,767.00 132,642.00
GPU                         117,291.58 160,767.00 132,642.00
Unique                      4,747.96   6,155.00   5,858.00
Parents                     844.18     484.00     417.00
Total                       5,992.00   6,609.00   6,299.00
New                          982.57     2,518.00   2,218.00
Visible Streaming Data Size (MB) 752.43     825.75     787.48
Streaming Pool Percentage    124.92     161.28     153.78
IO Request Size (MB)        0.25       0.69       0.18
```


Nanite - スタテック ディスプレイメント マッピング

実験的機能

- ・ テクスチャと軽量なメッシュで、非破壊でディテールを追加したNaniteメッシュを作成できる機能



Nanite - スタテック ディスプレイスメント マッピング

実験的機能

・使用方法

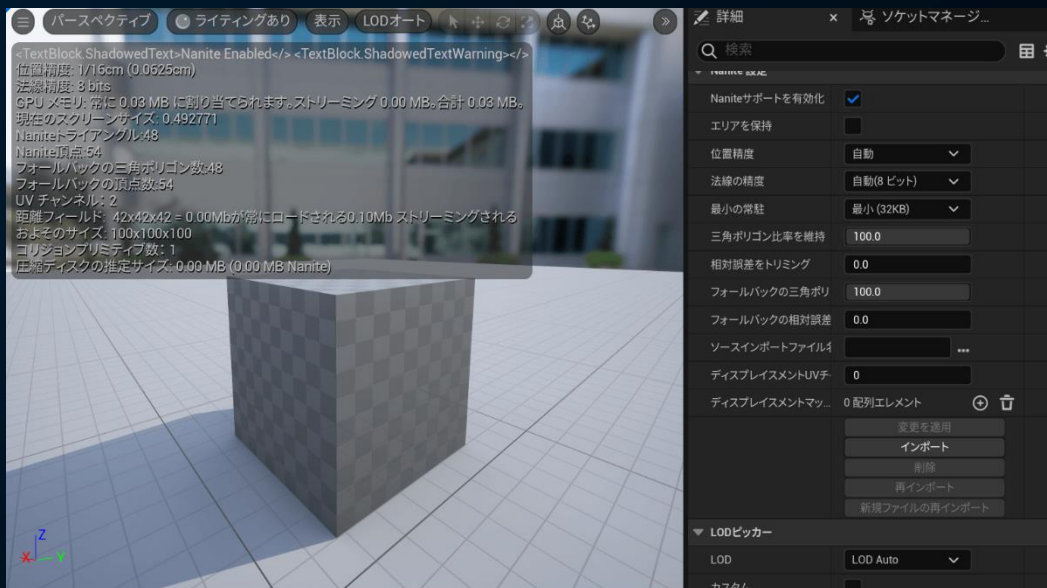
Nanite化したStatic Meshを開きNanite設定から設定を行う

1. 相対誤差をトリミングを設定
(分割が細かくなる)

2. テクスチャを設定

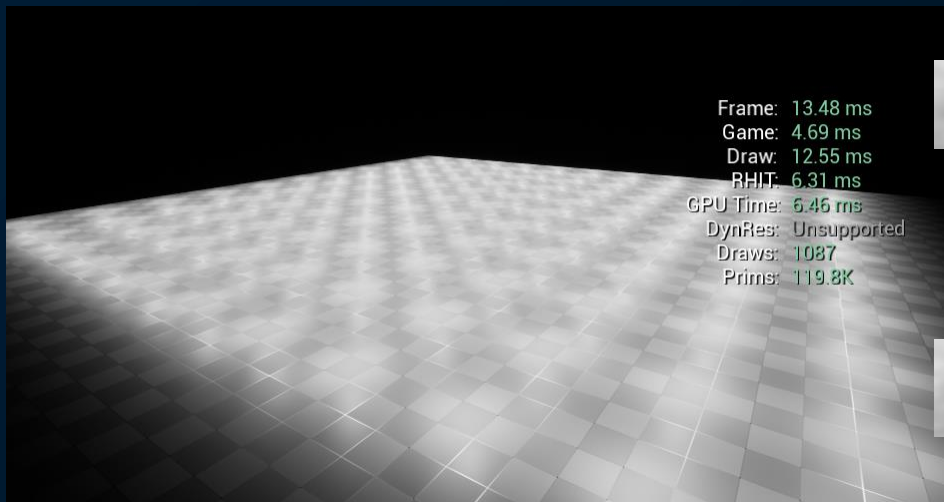
3. Magnitudeを調整
(押し出し量)

変更を適用で実行



VSM - シェドウに関する改善

- ・ **Distant Lights**と**Onepass Projection**がデフォルト有効化



GPU Time: 6.46 ms



両方を有効化

GPU Time: 4.32 ms

ポイントライト:100個配置
RTX4090にて計測

VSM - Distant LightsとOnepass Projection

・ Distant Lights

影響する領域が小さい遠くのライトの更新頻度を減らす
`r.shadow.virtual.distantlightmode 1`

更新数の調整

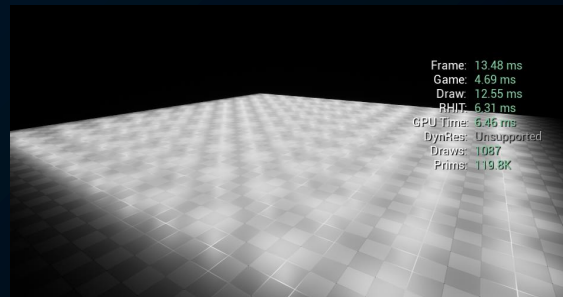
`r.Shadow.Virtual.MaxDistantUpdatePerFrame 1`

・ Onepass Projection

制限を満たすライトのシャドウプロジェクションをまとめて処理する
`r.Shadow.Virtual.OnePassProjection 1`

※制限については公式VSMドキュメントワンパス投影に記載

<https://docs.unrealengine.com/5.2/ja/virtual-shadow-maps-in-unreal-engine/>



TSR - Temporal Super Resolutionの改善

プラットフォームに依存しないテンポラルのアップスケール処理
例えば内部解像度はフルHDとしてレンダリングを行い4K解像度に

→各機能により
増大する**GPU負荷を削減**



TSR - Stat TSR

・ Stat TSRで表示される項目

TSR Feed

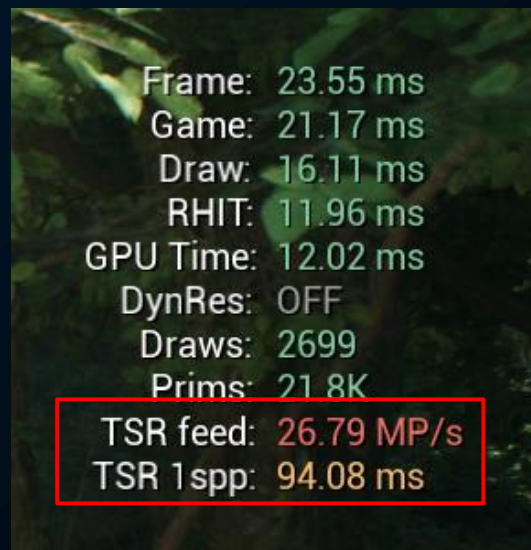
= 描画解像度の幅 * 描画解像度の高さ * フレームレート

※TSRに供給される情報量

TSR 1spp

= $1000 / (\text{スクリーン比率}^2 * \text{フレームレート})$

※最終的に出力したい1ピクセルを
表現するのに必要な時間



A screenshot of the Unreal Engine Stat TSR overlay, which displays various performance metrics in green text over a blurred game scene. The metrics include Frame, Game, Draw, RHIT, GPU Time, DynRes, Draws, and Prims. The last two metrics, TSR feed and TSR 1spp, are highlighted with a red rectangular box.

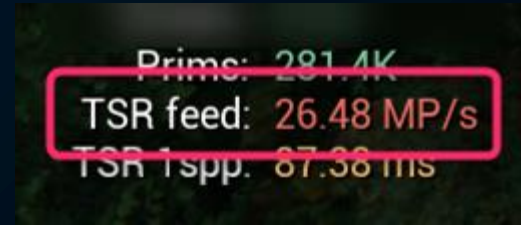
Frame:	23.55 ms
Game:	21.17 ms
Draw:	16.11 ms
RHIT:	11.96 ms
GPU Time:	12.02 ms
DynRes:	OFF
Draws:	2699
Prims:	21.8K
TSR feed:	26.79 MP/s
TSR 1spp:	94.08 ms

TSR - TSR Feed

- ・ TSRに供給される情報量
最終的な解像度ではなく実際の**描画解像度の情報量**を表す項目

例

4K (3840x2160)	50%	60hz	124.4 MP/秒
1080p (1920x1080)	50%	60hz	31.1 MP/秒



表示色は描画ピクセル数を元にした指標
921,600(1280 * 720)未満の場合:赤色
2,073,600(1920 * 1080) 未満の場合:オレンジ
2,073,600(1920 * 1080) 以上の場合:緑色

・まとめ

同じスクリーンパーセンテージ50%でも最終解像度が低い
(描画解像度が低い)と情報が少なくなってしまう

TSR - TSR 1spp

TSR 1spp (sample per pixel)

・最終的に出力したい解像度の1ピクセルに必要な時間

例：

1080p → 4K を想定する場合、
スクリーンパーセンテージは50% (2倍のスケール)
縦と横を考慮し4回のレンダリングが必要になる
Frame約 25 msの場合、**1spp は約 100 ms**

まとめ

アップスケールの比率を上げつつ
品質を保つためにはフレームレートが重要



表示色は1080p to 4K - 60hzを想定した指標
64 ms (16 ms * 4)未満の場合: **緑色**
128 ms (16 ms * 4 * 2)未満の場合: **オレンジ**
128 ms (16 ms * 4 * 2)以上の場合: **赤色**

TSR - 残像(ゴースティング)対策

- ・ TSRは過去フレームを蓄積させていく
主にVelocity(速度)情報を元に判断 → Velocityがないと誤った蓄積が行われる → **残像**

※対策(Velocityを書き出す)

各種マテリアルへの対応などを弊社篠山が記事にて公開

TSR時のWPOによる頂点移動の注意点(不透明オブジェクト編)

<https://dev.epicgames.com/community/learning/tutorials/9G0z/unreal-engine-tsr-wpo>

TSRによりTextureのスクロールで残像が出る際の簡易的な対処法(不透明マテリアル)

<https://dev.epicgames.com/community/learning/tutorials/zr8q/unreal-engine-tsr-texture>

SpeedTreeで作成した木がTSRによりGhostingを引き起こしてしまう場合の対処法

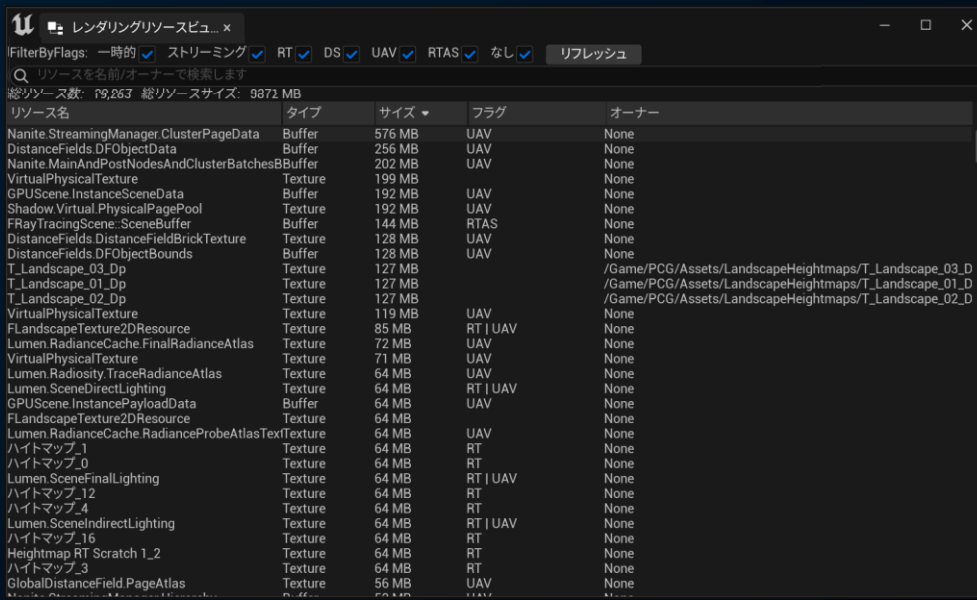
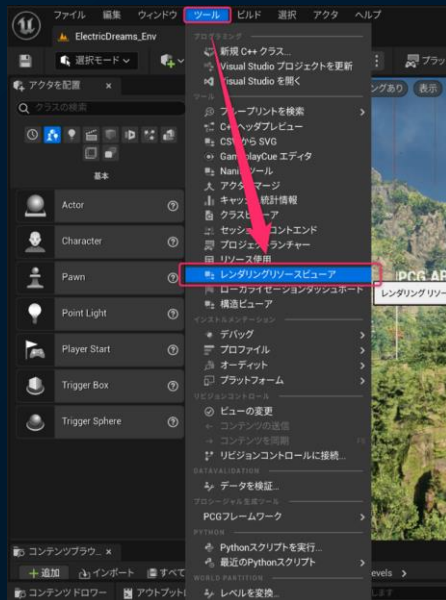
<https://dev.epicgames.com/community/learning/tutorials/vrkW/unreal-engine-speedtree-tsr-ghosting>

VAT(Vertex Animation Texture)がTSRによってブレる場合の簡易的な対処法

<https://dev.epicgames.com/community/learning/tutorials/4Onk/unreal-engine-vat-vertex-animation-texture-tsr>

Render Resource Viewer

- 描画に関わるリソースを一覧化するツール
[ツール] → [レンダリングリソースビューア] から起動



各種コンソール上でみたい場合は、`rhi.DumpResourceMemory`コマンドでコンソールに出力

可変レート シェーディング

実験的機能

- ・ 直前のフレームを解析し領域ごとに粗いシェーディングを行い高速化を目指す
※2x2や4x4毎に1つのピクセルシェーダーを走らせる

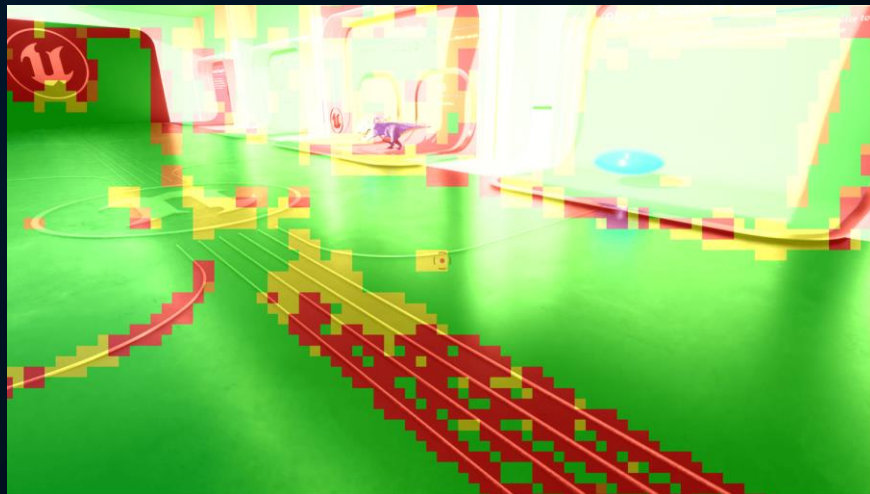
- ・ **機能の有効化**

r.VRS.ContrastAdaptiveShading 1

- ・ **可視化**

r.VRS.Preview 1

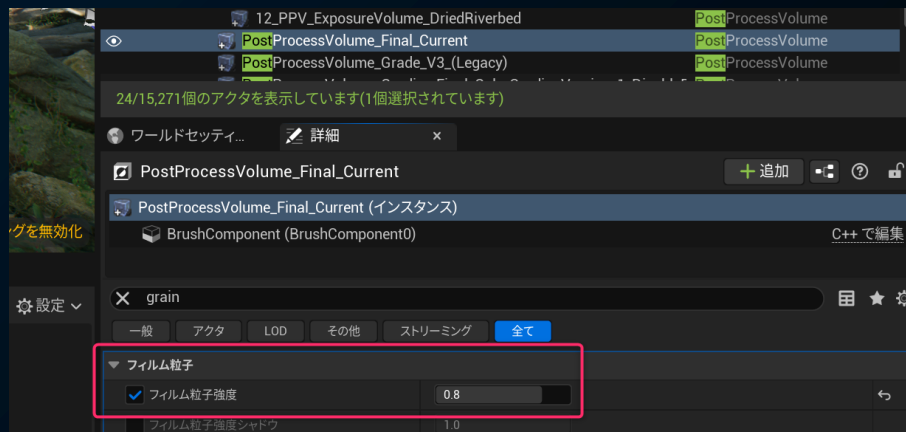
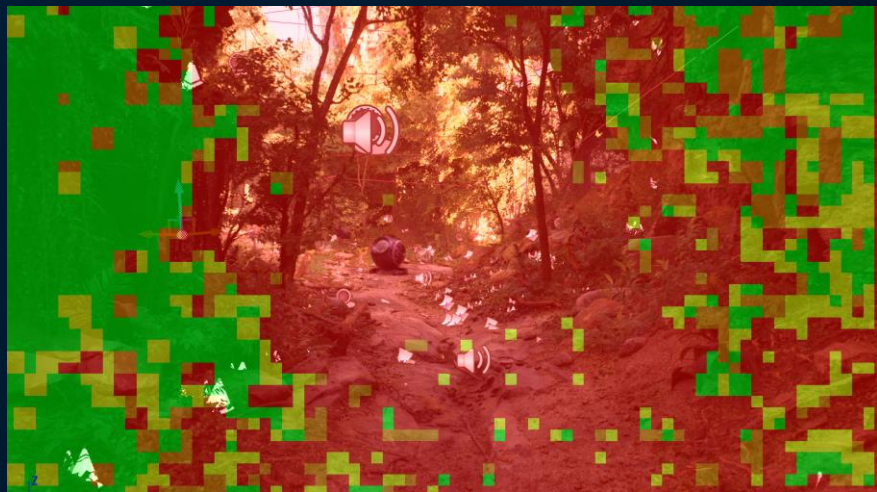
現時点ではパフォーマンス上、
劇的に改善することは確認できておらず、
実験的にお試し頂ければと思います。



可変レート シェーディング

実験的機能

- ・ Electric Dreams
Film Grain(フィルム粒子)効果がある場合利用できない
Post Process Volumeから無効化すると利用可





PCG

Procedural Content Generation Framework

背景に対して今後求められる要素は？



広大

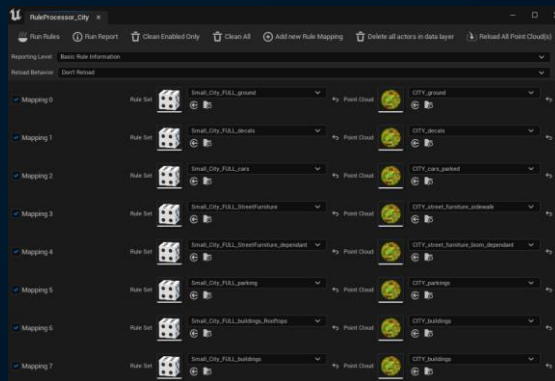
高密度

高速イテレーション

CitySample と RuleProcessor

City サンプル クイック スタート - Unreal Engine 5 で街と高速道路を生成する

Houdini で生成されたプロシージャルなデータを使用して Unreal Engine 5 で街を作成するためのガイドです。



PCG フレームワーク

目的

- ✔ **エンジン内蔵**
- ✔ プロシージャル生成
- ✔ 非破壊編集
- ✔ ノードベース & 拡張性
- エディタ/ランタイムで動作
- **Experimental**



PCGに関する講演



New Tools for Building Photoreal Worlds in Unreal Engine 5.2 (日本語字幕)

<https://www.youtube.com/watch?v=zxpoY1TfkY>



Procedural Tools In UE5 (日本語字幕)

<https://www.youtube.com/watch?v=aoCGLW53fZg>



Deep Dive into the Electric Dreams Project | Inside Unreal

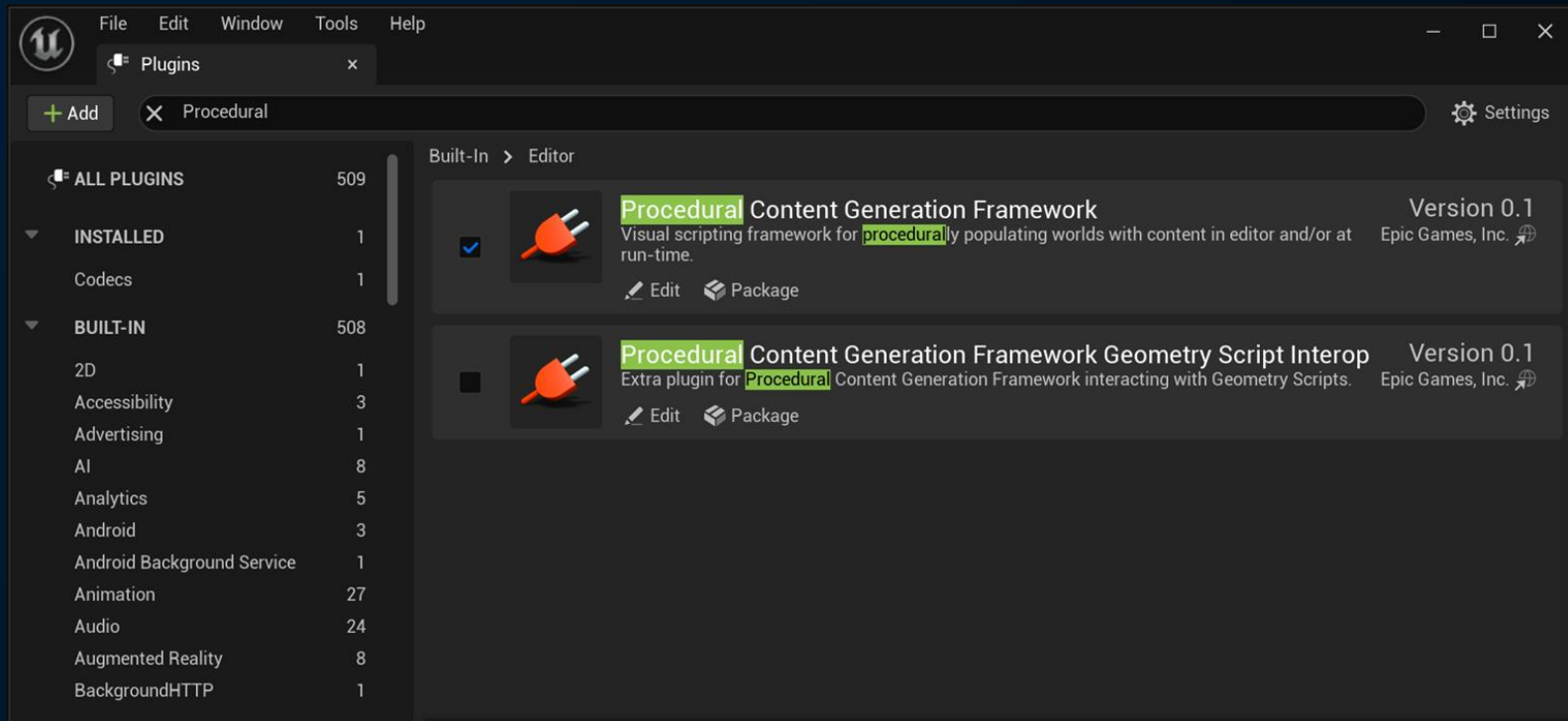
<https://www.youtube.com/watch?v=6JUfisUhm68>

Electric Dream environment サンプル



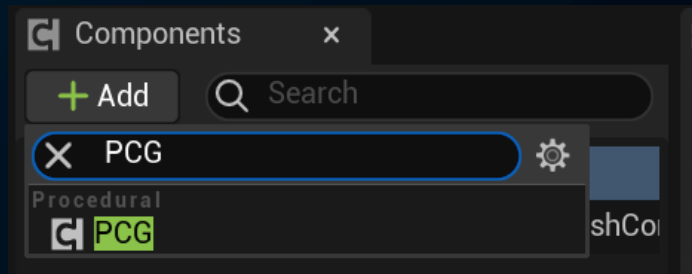
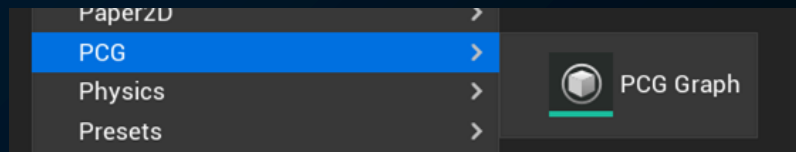
<https://www.unrealengine.com/ja/electric-dreams-environment>

プラグインの有効化



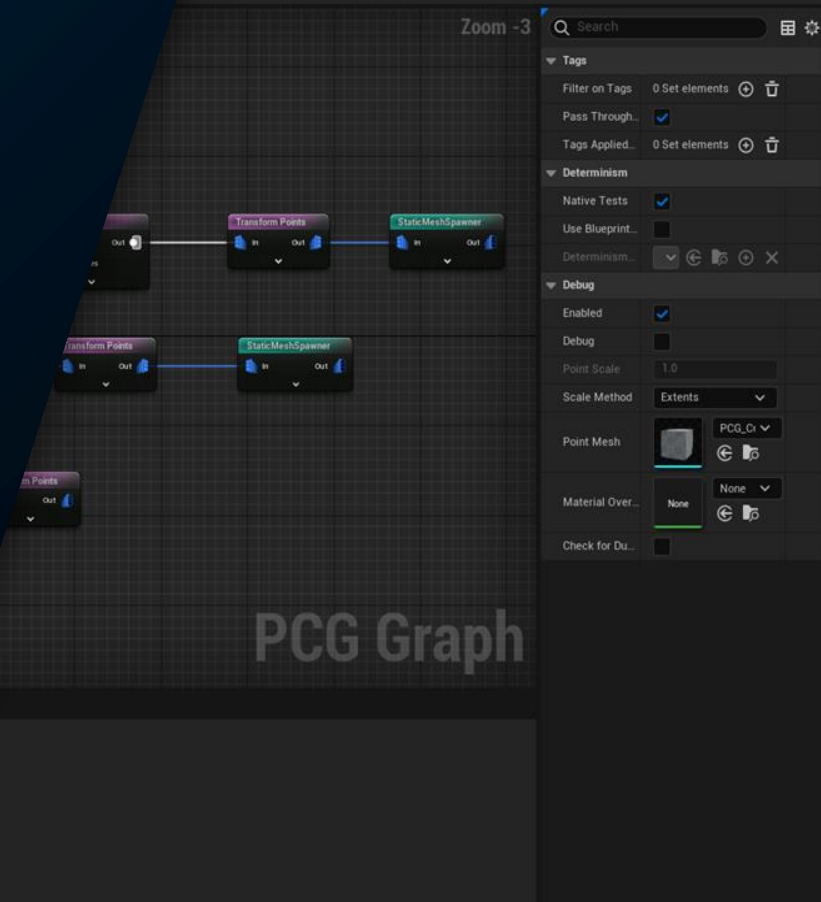
PCG フレームワーク

- PCGGraph
 - マテリアルのようなノードグラフ
- PCGComponent
 - PCGGraphを実行するコンポーネント
- PCGVolume
 - PCGComponentとボリュームを持ったアクター



PCGGraphのノード構造

- データの取得
- ポイントの生成
- ポイントの加工
 - フィルタ
 - 変形
 - 演算
- 配置



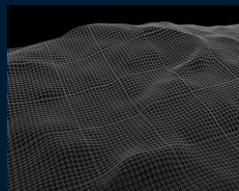
PCGGraphの構造

空間データ

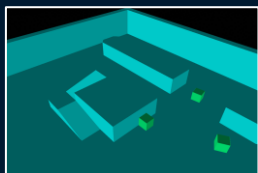
ポイント生成

加工

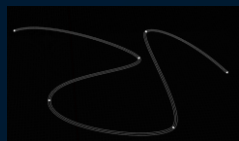
配置



Landscape



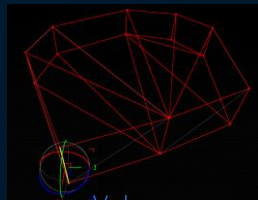
Collision



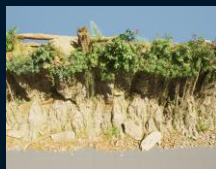
Spline



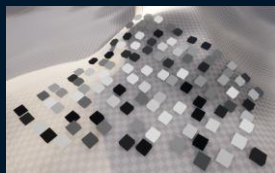
Texture



Volume



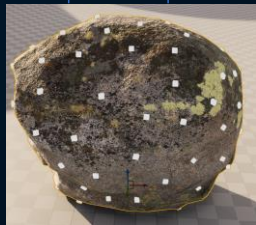
PCGSetting



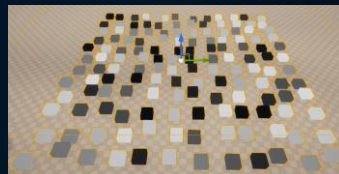
SurfaceSampler



SplineSampler



MeshSampler



Filter



Pruning



Difference

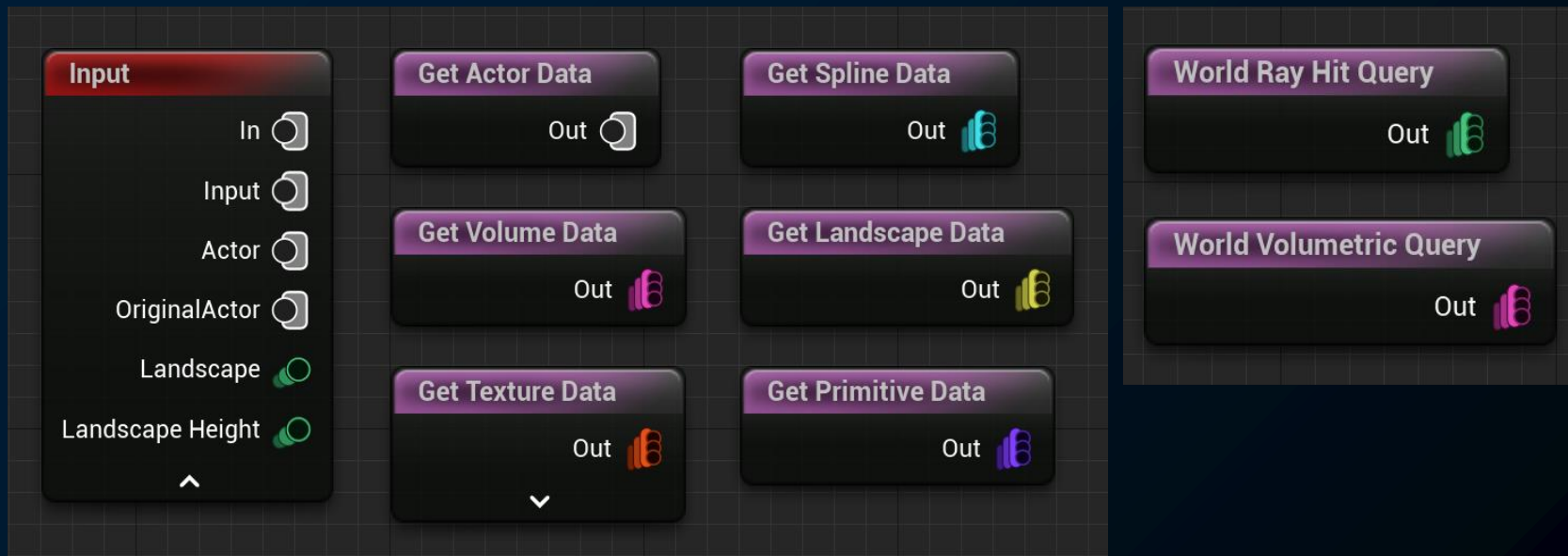


StaticMesh

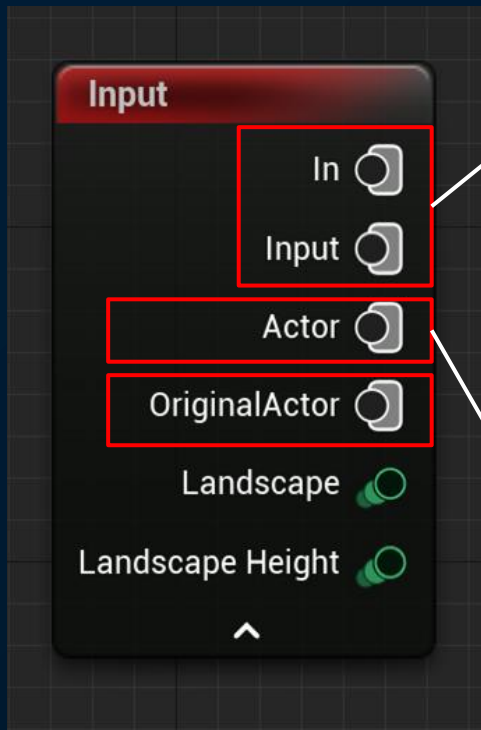


Actor

代表的な空間データ取得ノード



インプット



PCGコンポーネントのInputTypeにより変化
InputType=Actor
→ アクターと同じ
InputType=Landscape
→ アクターとランドスケープを交差したもの

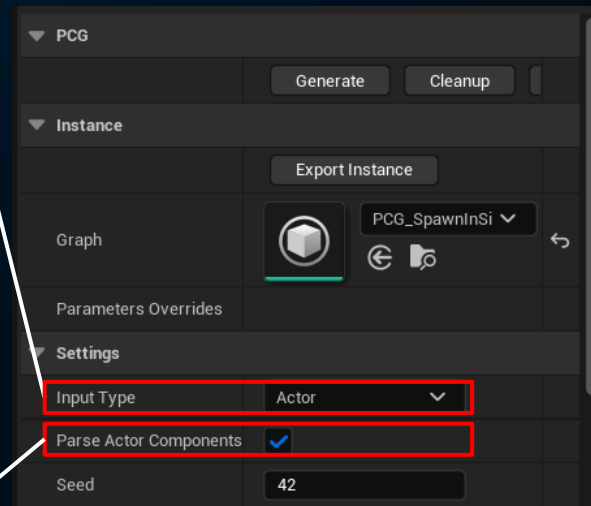
PCGコンポーネントのInputTypeにより変化
コンポーネントがパーティション化されている
→ パーティションアクター

ランドスケーププロキシアクターにPCGが刺さっている
→ ランドスケープ

アクターがボリュームをルートに持っている
→ ボリューム

上記に当てはまらず ParseActorComponent:True
→ 全プリミティブをユニオンしたもの

上記全てに当てはまらず ParseActorComponent:False
→ アクターのバウンズ

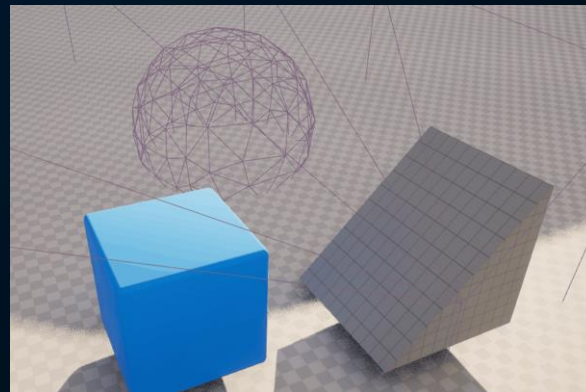
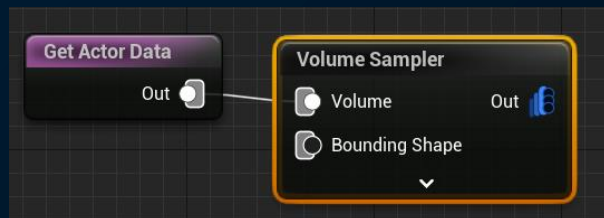


詳しくは
[UPCGComponent::CreateActorPCGDataCollection](#)
を参照してください

代表的なポイント生成ノード

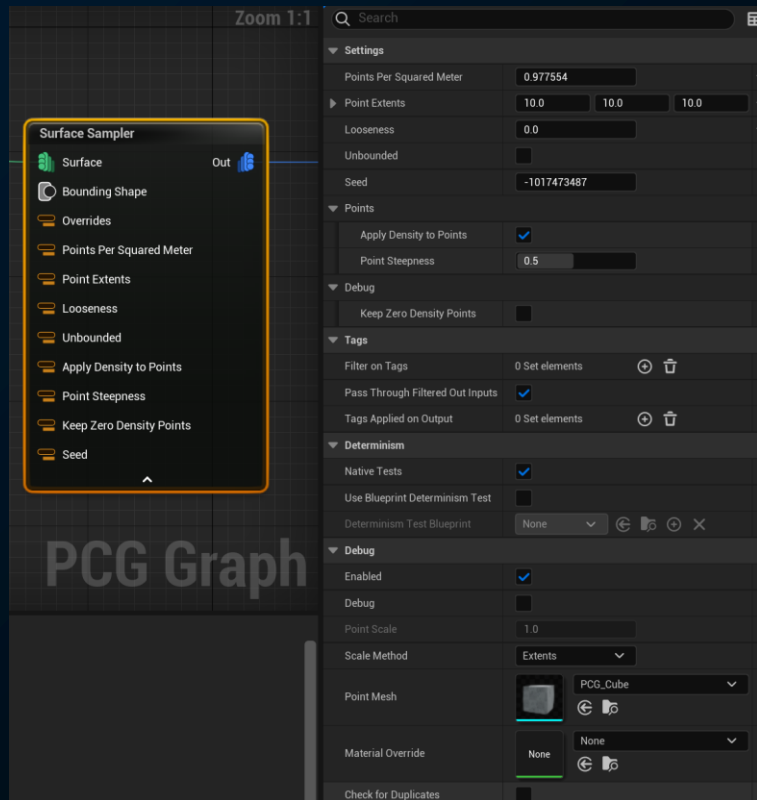
▼ Sampler

BP_Element_MeshToPointsWithColor
Copy Points
Mesh Sampler
Point Sampler
Spline Sampler
Surface Sampler
Volume Sampler

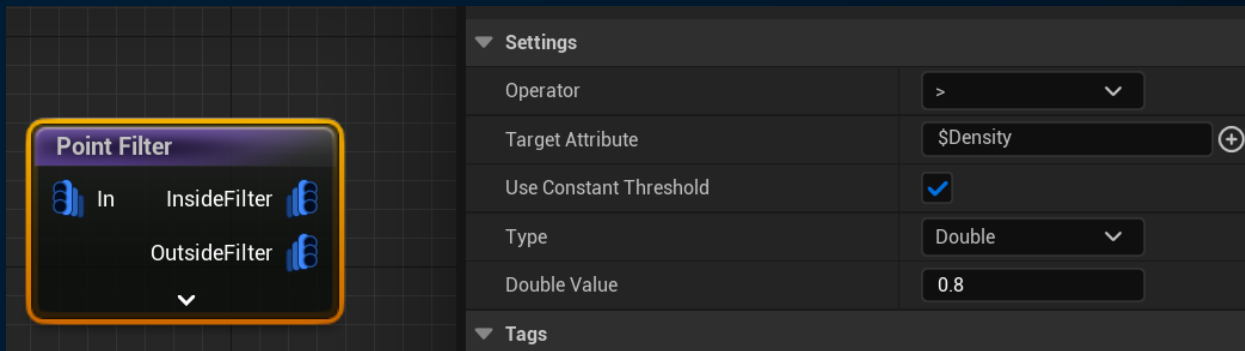


SurfaceSampler

- Points Per Squared Meter
 - 面積に対するポイントの最大数
- Point Extents
 - バウンズの大きさ
- Looseness
 - $\text{バウンズ} * \text{Looseness}$ でセルサイズが決まる
 - 0 の時 Bounds = CellSize となり
ぎっしり敷き詰められる
- Unbounded
 - バウンズを無視してワールド全体を対象にする



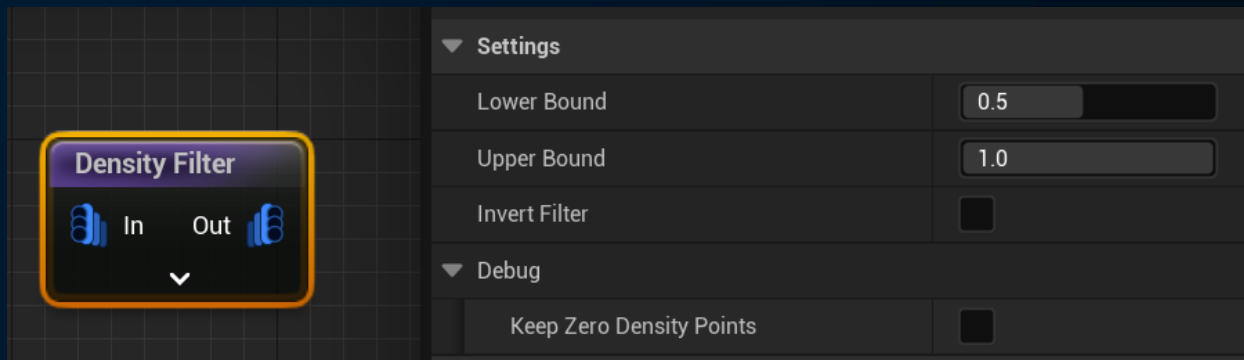
代表的な加工ノード (フィルタ1)



The image shows a 'Point Filter' node in the Unreal Engine visual scripting system. The node is highlighted with a yellow border and contains three filter options: 'In', 'InsideFilter', and 'OutsideFilter'. To the right, the settings panel is visible, showing the following configuration:

Settings	
Operator	>
Target Attribute	\$Density
Use Constant Threshold	<input checked="" type="checkbox"/>
Type	Double
Double Value	0.8

Below the settings, there is a 'Tags' section which is currently empty.



The image shows a 'Density Filter' node in the Unreal Engine visual scripting system. The node is highlighted with a yellow border and contains two filter options: 'In' and 'Out'. To the right, the settings panel is visible, showing the following configuration:

Settings	
Lower Bound	0.5
Upper Bound	1.0
Invert Filter	<input type="checkbox"/>

Below the settings, there is a 'Debug' section with the following option:

Debug	
Keep Zero Density Points	<input type="checkbox"/>

代表的な加工ノード (フィルタ2)

The image shows a PCG Graph in Unreal Engine. A 'Get Primitive Data' node is connected to a 'FilterByTag (Keep)' node. The 'FilterByTag (Keep)' node is highlighted with a yellow box. To the right, the configuration panel for this node is visible, showing the 'Settings' section with 'Operation' set to 'Keep Tagged' and 'Selected Tags' set to 'COMPONENTTAG_1, COMPONENTTAG_3'. A white box highlights the 'Selected Tags' field. Below the graph, a 'Find' window is open, showing a list of primitive data outputs. The first two entries are highlighted with a blue box, and the entire 'Find' window is outlined in red.

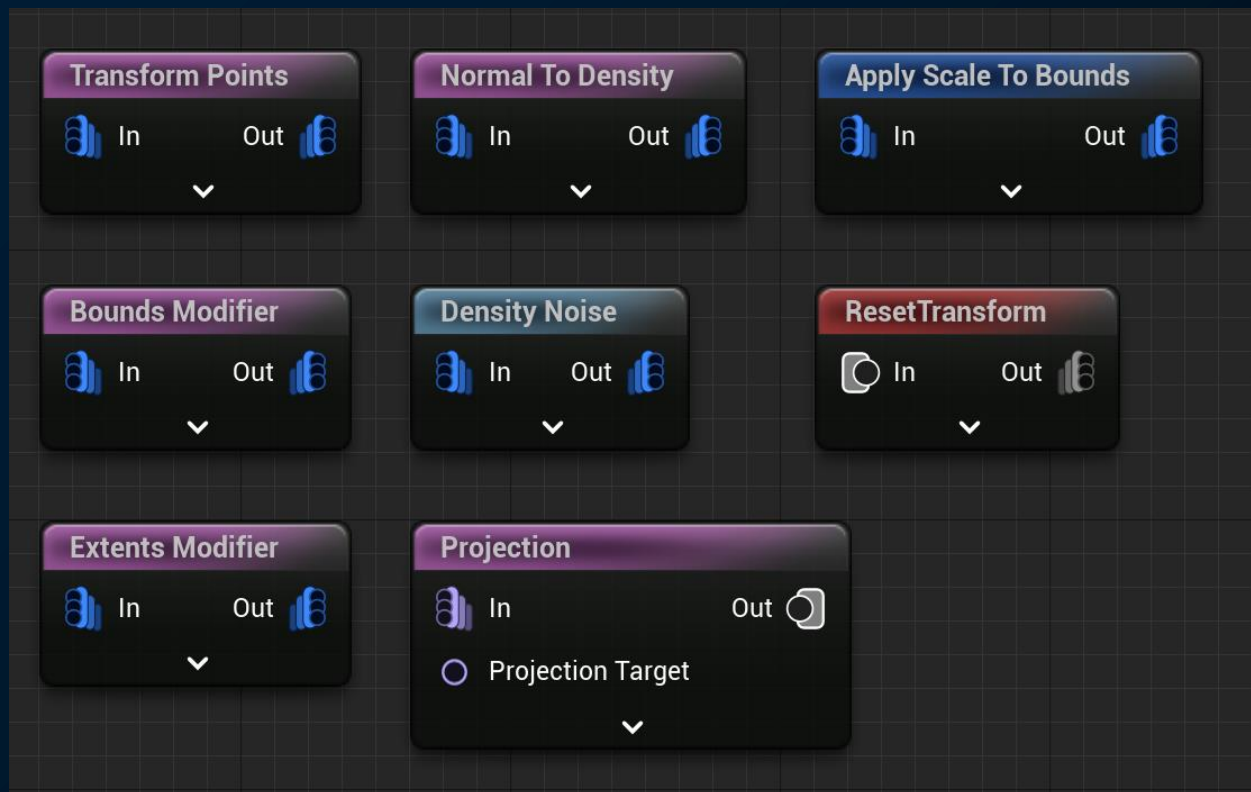
FilterByTag (Keep) Settings:

- Operation: Keep Tagged
- Selected Tags: COMPONENTTAG_1, COMPONENTTAG_3

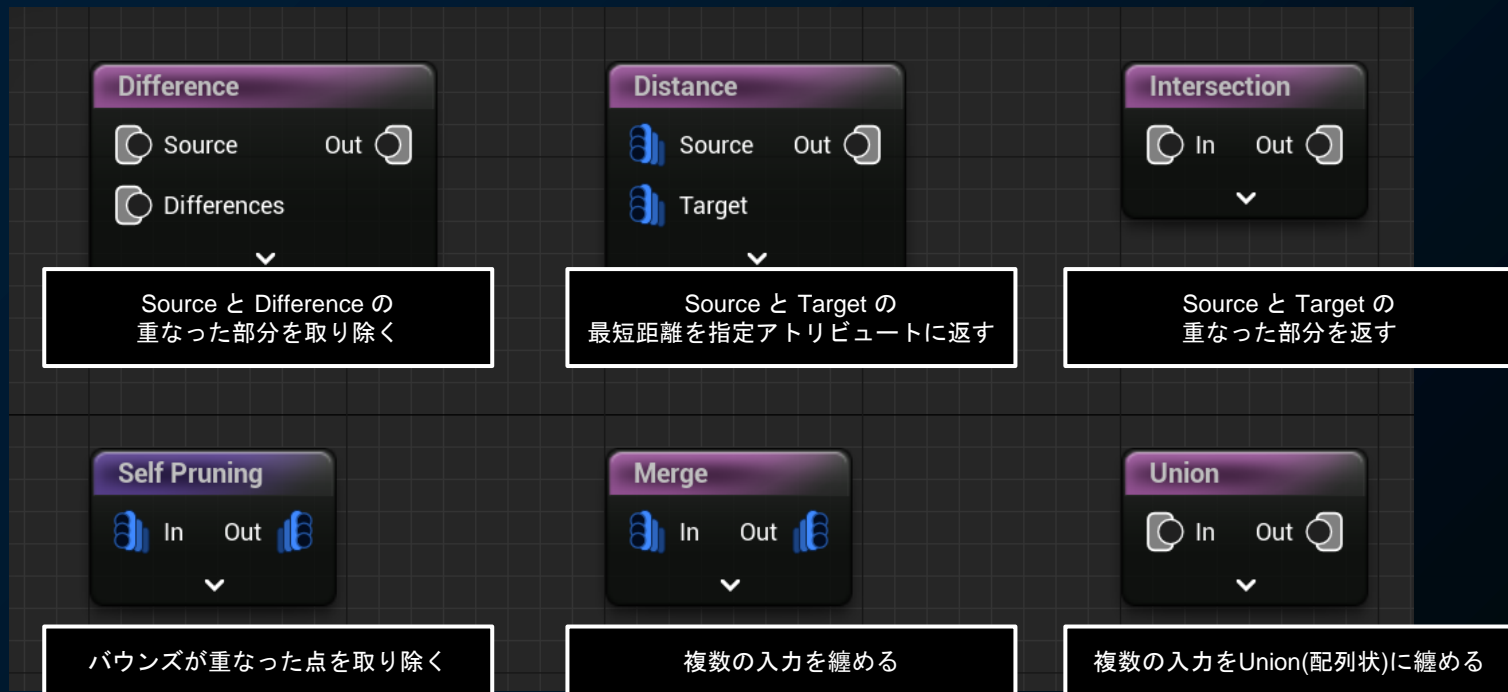
Find Window:

Find	Attributes	x
[0] Out - PCGPrimitive Data: (COMPONENTTAG_1, COMPONENTTAG_2, ACTORTAG_EXCLUDE)		
[0] Out - PCGPrimitive Data: (COMPONENTTAG_1, COMPONENTTAG_2, ACTORTAG_EXCLUDE)		
[1] Out - PCGPrimitive Data: (COMPONENTTAG_3, ACTORTAG_EXCLUDE)		

代表的な加工ノード (ポイント)



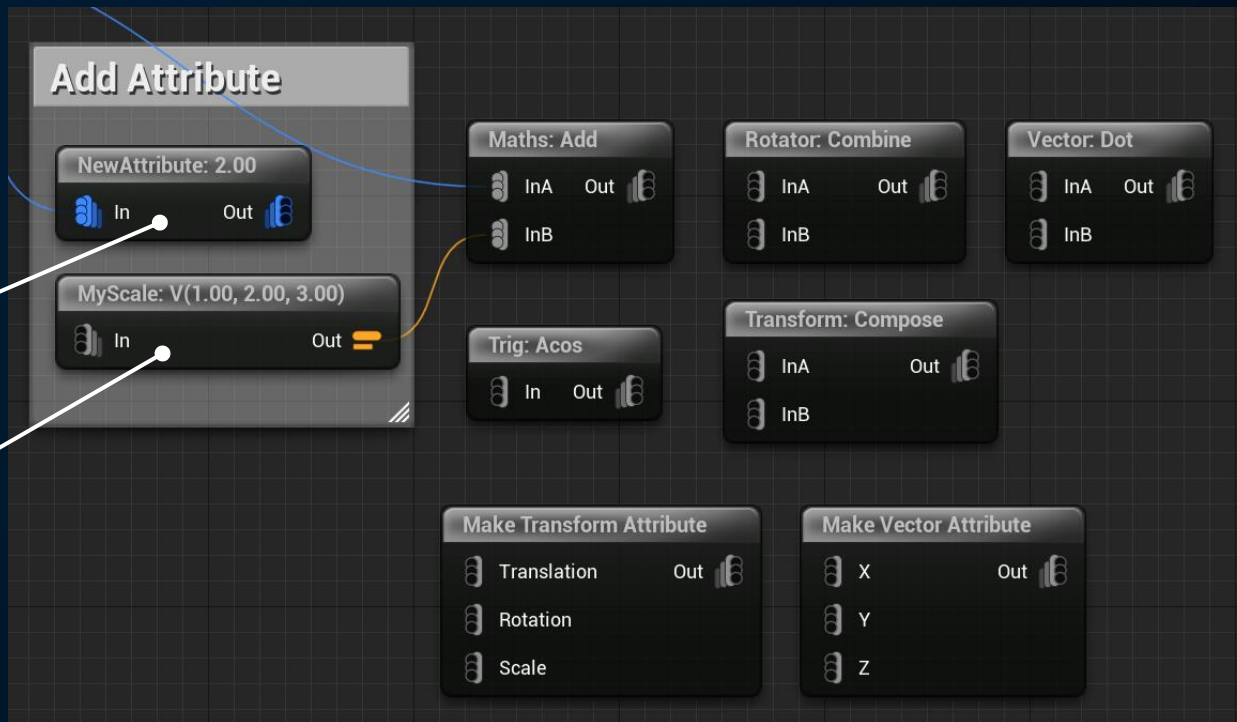
代表的な加工ノード (点群に対する演算)



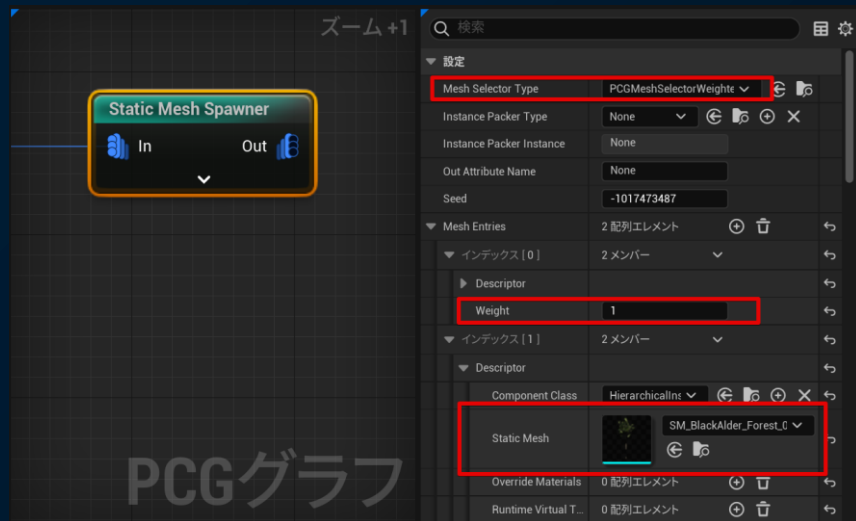
代表的な加工ノード (変数演算)

全てのポイントに
変数を追加

定数を作成



代表的な生成ノード



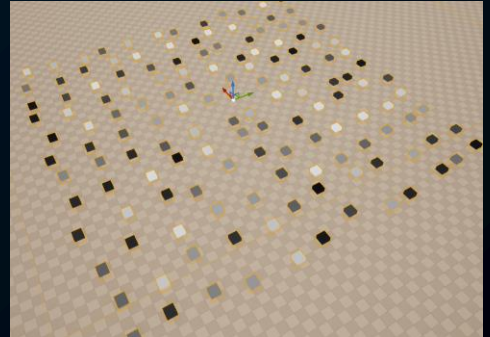
TIPS

- デバッグ機能
- アンリンク(スタンプ)
- アクターのパラメータを利用する
- プロパティとアトリビュート
- PCGElement
- LandScapeLayer
- Tag
- PCG Settings



デバッグ

The screenshot displays the Unreal Engine PCG Editor interface. The main workspace shows a graph with two 'Bounds Modifier' nodes and an 'Output' node. A context menu is open over the second 'Bounds Modifier' node, with the 'Debug' option highlighted in red. The 'Settings' panel on the right shows the 'Debug' section with the 'Debug' checkbox checked, also highlighted in red. The top toolbar includes options like 'Pause Regen', 'Force Regen', and 'Cancel Execution'. The top status bar indicates 'BoundAndScale (selected) / PCG Component'.



デバッグ

● ScaleMethod

● Relative

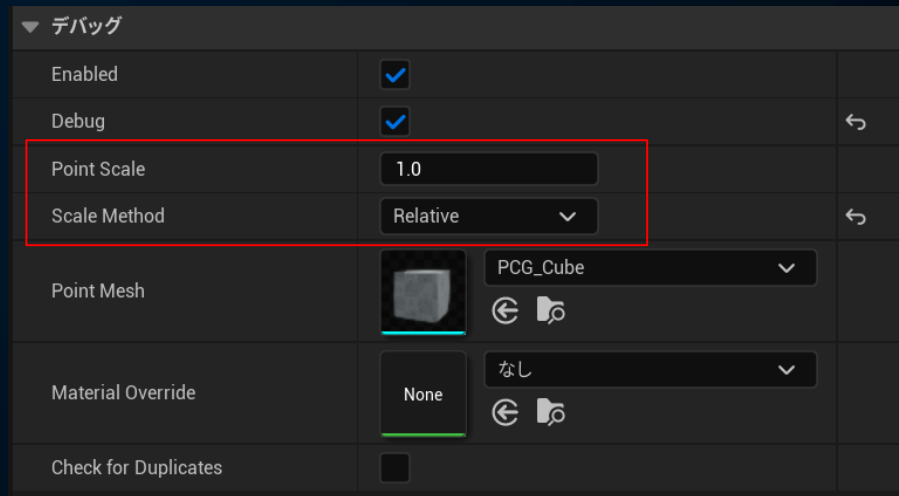
● **TransformのScale** * Point Scale

● Absolute

● Point Scale

● Extents

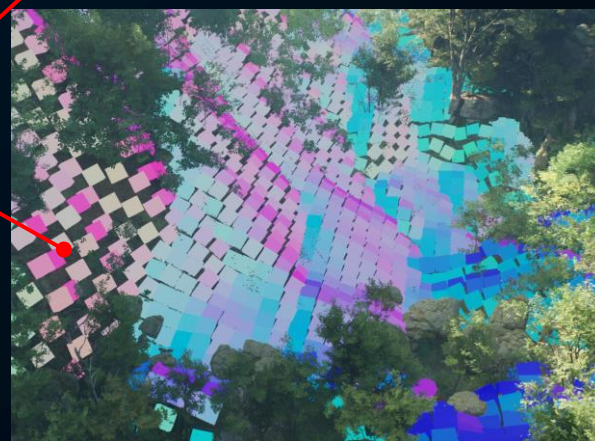
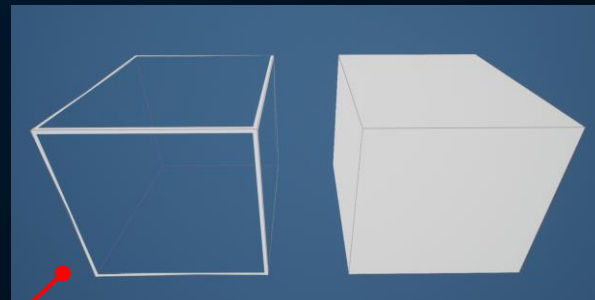
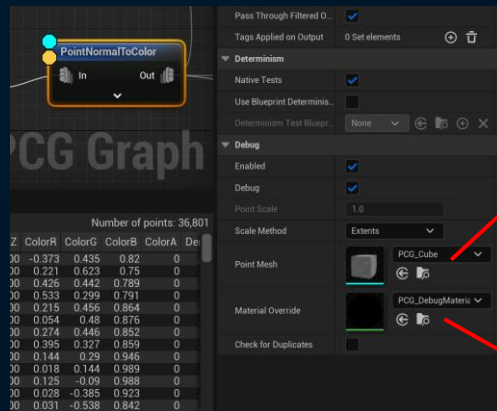
● **Boundsの大きさ**



デバッグ

- PointMesh

- Material Override



デバッグ

GENERAL

- Delete DELETED
- Cut CTRL+X
- Copy CTRL+C
- Duplicate CTRL+D

NODE ACTIONS

- Enable E
- Debug D
- Debug Only Selected CTRL+ALT+D
- Disable Debug on all nodes ALT+D
- Inspect A
- Break Node Link(s)
- Collapse into Subgraph ALT+J
- Export nodes to AssetData
- Convert to standalone Nodes

ORGANIZATION

- Set Node Color
- Alignment >

COMMENT GROUP

- Create Comment from Selection C

DETERMINISM

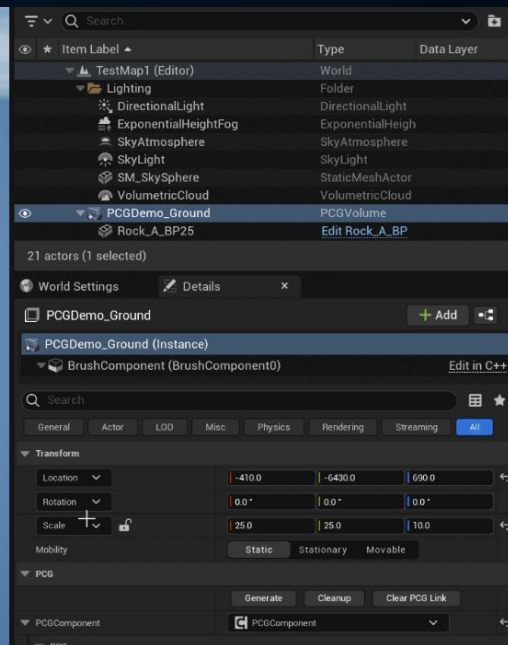
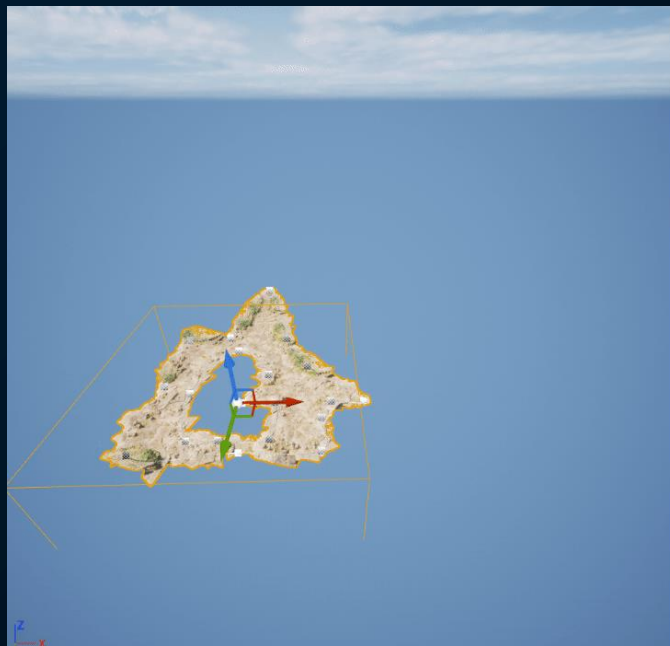
- Validate Determinism on Selection ALT+T

The screenshot shows the Unreal Engine PCG Graph editor. The 'BoundAndScale' component is selected, and the 'Output' node is connected to it. The 'Bounds Modifier' node is highlighted with a yellow border. The output data table is visible at the bottom of the graph.

Index	PositionX	PositionY	PositionZ	RotationX	RotationY	RotationZ	ScaleX	ScaleY	ScaleZ	BoundsMinX	BoundsMinY	BoundsMinZ	BoundsMaxX	BoundsMaxY	BoundsMaxZ	Number of points: 144
0	-5.909	0	-5.361	919	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
1	-5.538	9	-5.375	321	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
2	-5.169	9	-5.320	876	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
3	-4.677	4	-5.227	849	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
4	-4.249	17	-5.312	388	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
5	-3.810	5	-5.293	243	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
6	-3.501	5	-5.219	656	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
7	-3.117	8	-5.330	411	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
8	-2.631	1	-5.212	429	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
9	-2.351	8	-5.243	106	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
10	-1.900	5	-5.243	954	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
11	-1.496	2	-5.328	043	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
12	-1.082	35	-5.698	663	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
13	-0.571	8	-5.749	411	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
14	-0.025	9	-5.650	426	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
15	-4.662	3	-5.734	29	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
16	-4.218	1	-5.673	276	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
17	-3.808	9	-5.708	386	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
18	-3.509	9	-5.674	335	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
19	-3.126	7	-5.770	005	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
20	-2.722	9	-5.609	507	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400
21	-2.322	9	-5.623	300	0	-0.225	0.225	0	1	1	1	-400	-400	-400	400	400

PCGリンクをクリア

- PCGによる生成物を別アクターに切り離す

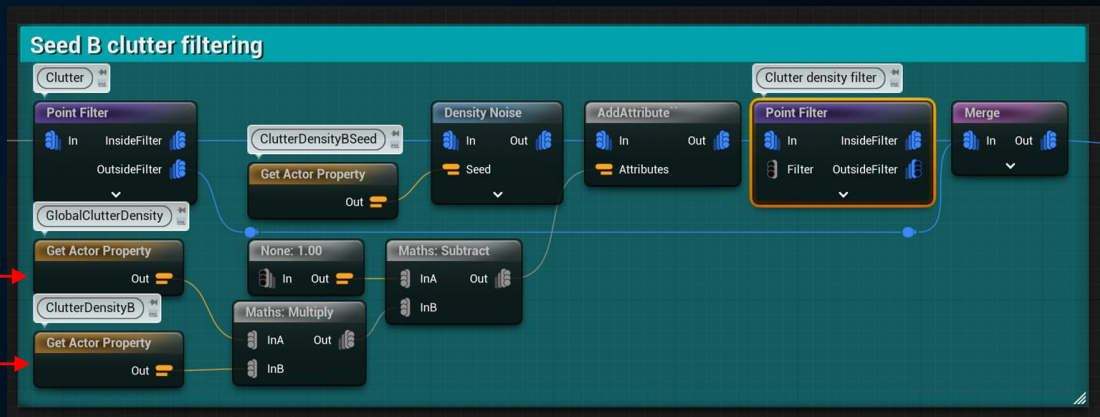
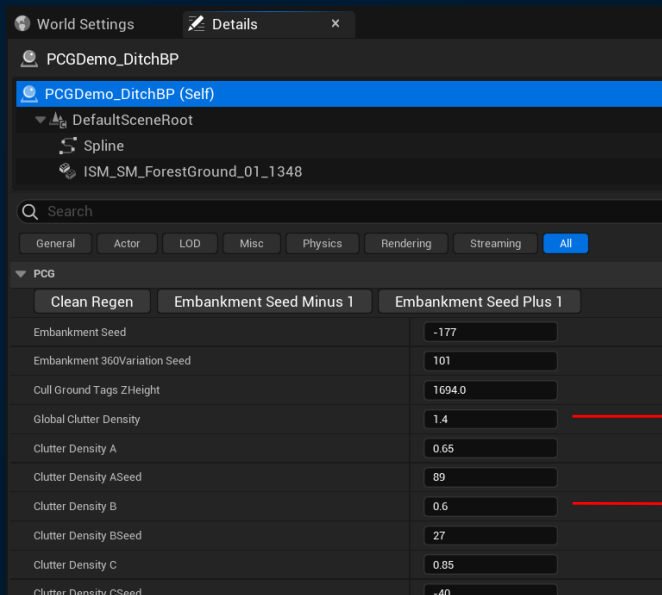


ランタイムでの動作

- PCGは「リンクをクリア」したものの以外はランタイムで動作
- クック時に静的に生成
 - 今後の改善を予定

アクターのパラメーターを利用する

- アクターのプロパティを読み出す
- シード値の微調整など



プロパティとアトリビュート

名前	記述方法	意味
Position	\$Position.X	位置(vec3)
Rotation	\$Rotation.Y	角度(Rotator)
Scale	\$Scale.Z	スケール(vec3)
Density	\$Density	密度値(float)
BoundsMin/Max	\$BoundsMin.X	バウンズの大きさ(vec3)
Color	\$Color.X	色(vec4)
Steepness	\$Steepness	密度関数の傾斜(float)
Seed	\$Seed	乱数シード(int64)
ユーザーアトリビュート	MyAttribute	※ \$を付けない

プロパティとアトリビュート

ズーム 1:1

PCGグラフ

属性 x 決定方法

[0] Out - Attribute Set Value: 14.52 メタ

インデックス Value

014.524

設定

Output Attribute Name Value

Type Double

Double Value 14.523942

タグ

Filter on Tags 0 セットエレメント

Pass Through Filter...

Tags Applied on Output 0 セットエレメント

Determinism

Native Tests

Use Blueprint Determinism Test

Determinism Test Behavior Nc

デバッグ

Enabled

Debug

Point Scale 1.0

Scale Method Extents

ズーム +1

PCGグラフ

属性 x 決定方法

[0] Out - PCGPoint Data: (Creation) Mat

インデックス	PositionX	PositionY	PositionZ	F
0	590	870	52.414	
1	890	870	18.281	
2	290	1,170	42.938	
3	590	1,170	30.203	
4	890	1,170	7.578	

設定

Operation Multiply

入力

Input Source 1 \$Density

Input Source 2 Value

出力

Output Target \$Scale.Z

タグ

Filter on Tags 0 セットエレメント

Pass Through Filter...

Tags Applied on Output 0 セットエレメント

Determinism

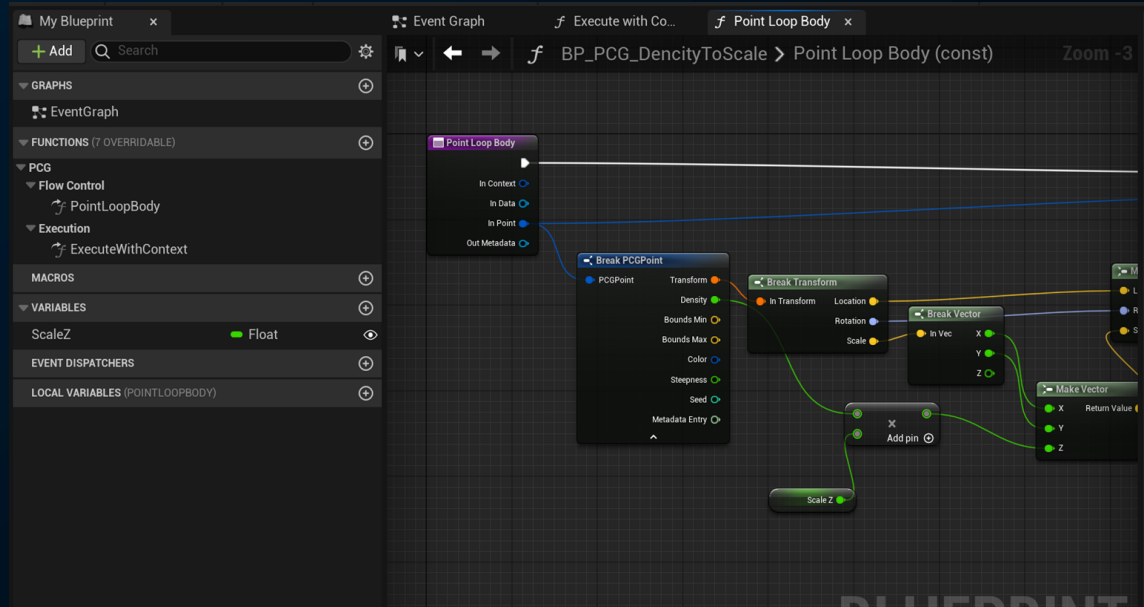
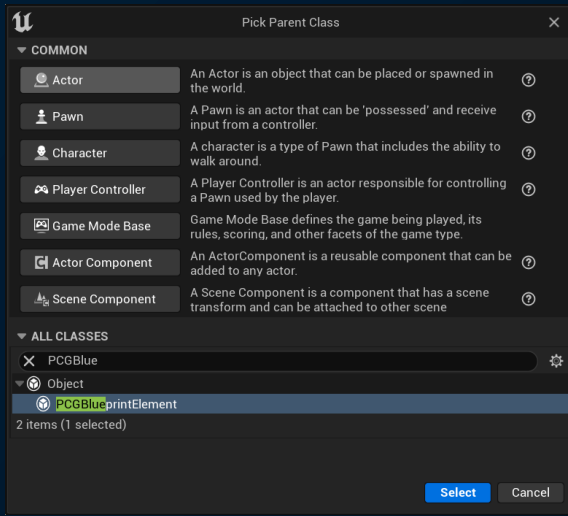
Native Tests

Use Blueprint Determinism Test

Determinism Test Behavior Nc

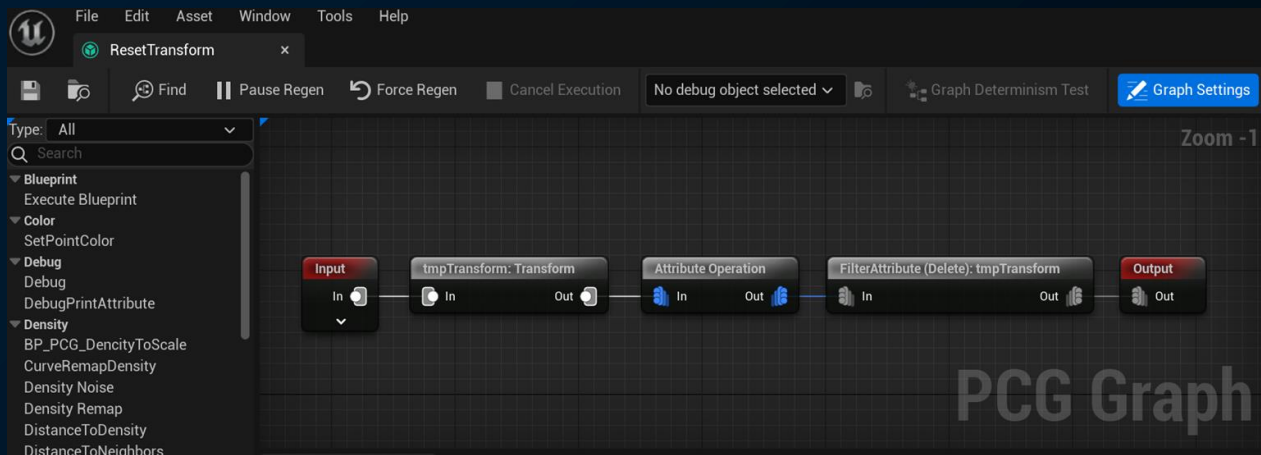
PCG Element

- PointLoopBodyはマルチスレッド動作
- ブループリントノードを使って点群内の各ポイントを加工

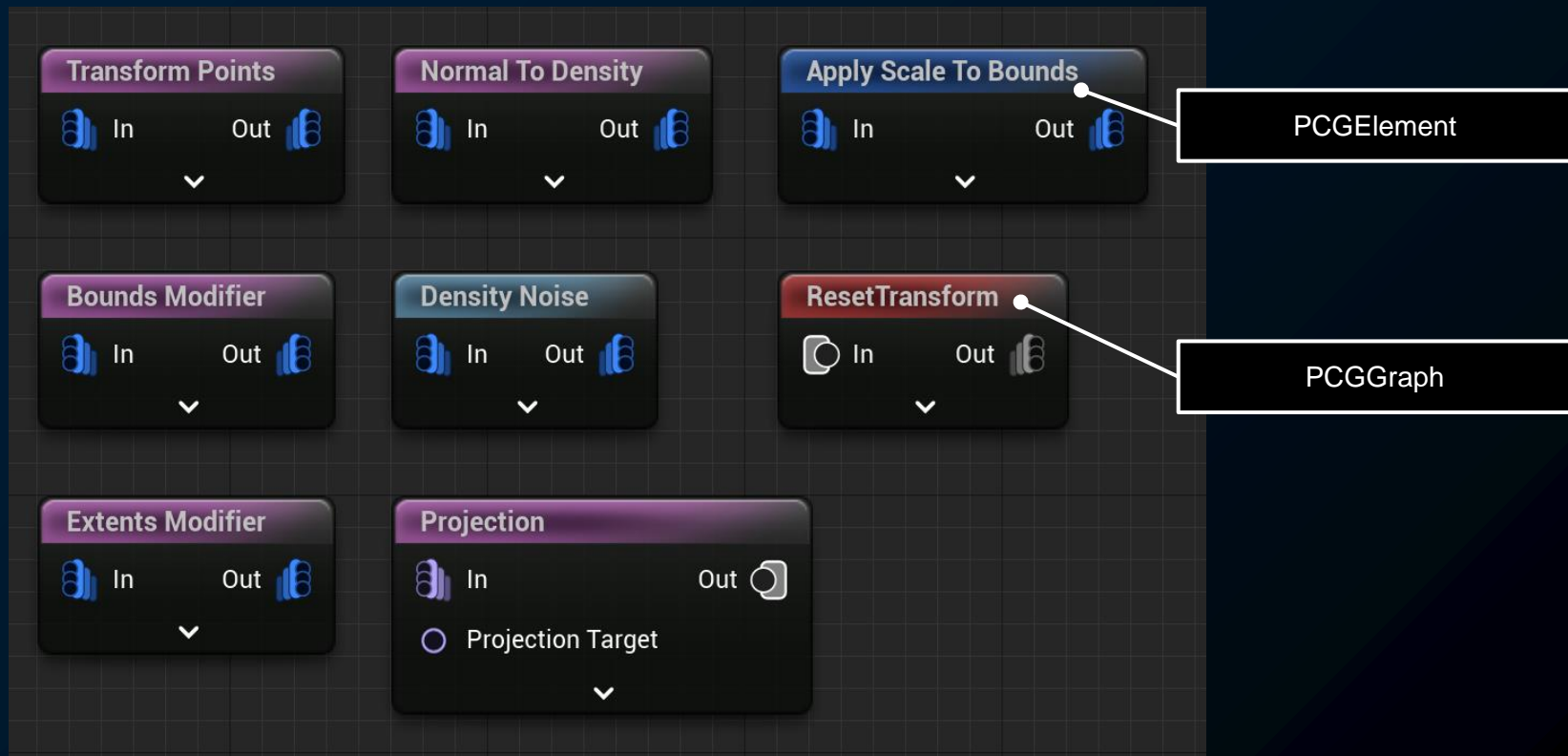


PCGGraphのマクロ的な利用

- Outputノードに繋がっているPCGGraphはマクロ的に利用できる
- 単純にインライン展開されるような動作
- InputとOutputにカスタムピンを追加可能

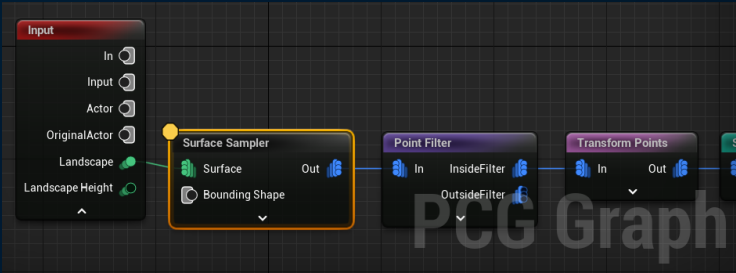


PCGElementとPCG呼び出し



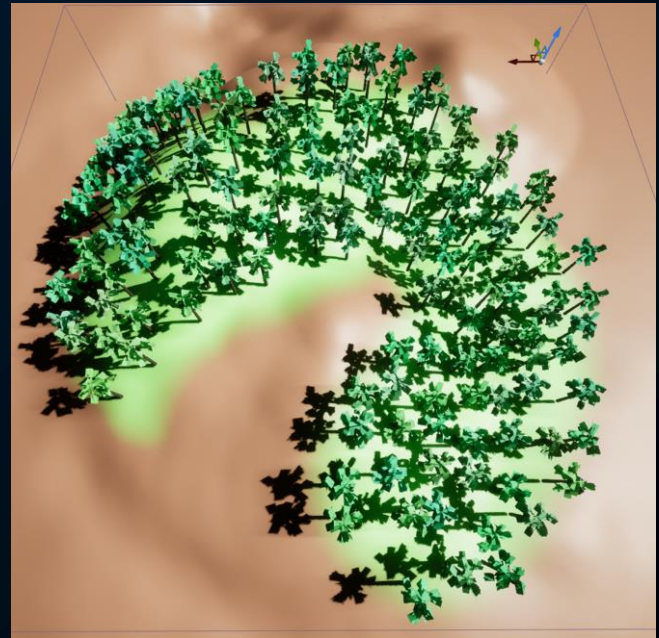
Landscape layer

5.2では 1x1 Sections Per Componentのみ対応
2x2 section対応 は5.3から(CL-25796774)



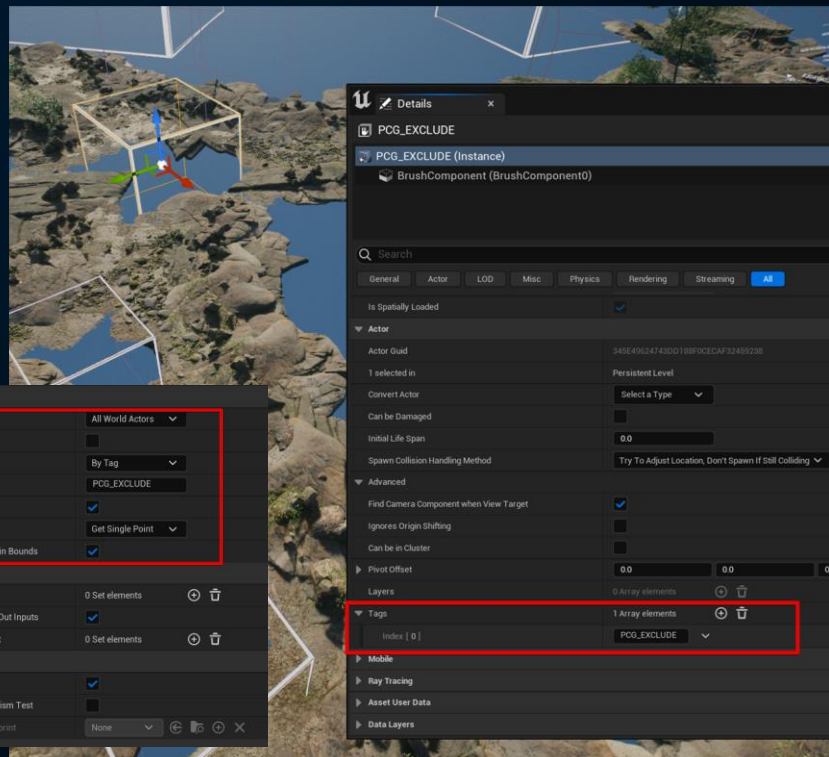
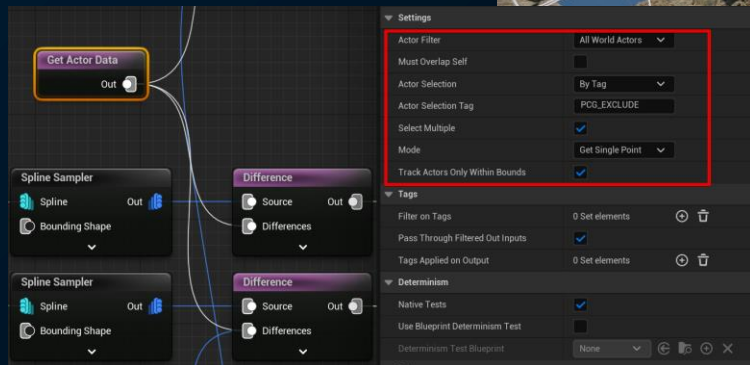
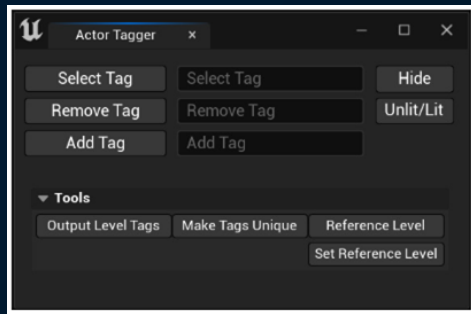
The screenshot shows a PCG Graph workflow. The 'Surface Sampler' node is highlighted with a yellow box. It has an 'Input' section with 'Landscape' and 'Landscape Height' checked. The 'Point Filter' node is connected to the 'Surface Sampler' and has 'InsideFilter' and 'OutsideFilter' options. The 'Transform Points' node is connected to the 'Point Filter' and has a dropdown menu. The 'PCG Graph' watermark is visible in the background.

BoundsMaxY	BoundsMaxZ	ColorR	ColorG	ColorB	ColorA	Density	Steepness	Seed	Base	Grass
100	100	1	1	1	1	0.465	0.5	-2.0	1	0
100	100	1	1	1	1	0.517	0.5	-234	1	0
100	100	1	1	1	1	0.329	0.5	351	1	0
100	100	1	1	1	1	0.763	0.5	1,57	0.782	0.218
100	100	1	1	1	1	0.134	0.5	-1,9	0.256	0.744
100	100	1	1	1	1	0.421	0.5	-1,6	0.042	0.958
100	100	1	1	1	1	0.593	0.5	5,88	0.478	0.522
100	100	1	1	1	1	0.399	0.5	1,18	0.912	0.088
100	100	1	1	1	1	0.441	0.5	1,31	1	0
100	100	1	1	1	1	0.872	0.5	-1,93	1	0
100	100	1	1	1	1	0.215	0.5	-1,9	1	0
100	100	1	1	1	1	0.774	0.5	101	1	0



Tag

- 除外設定などにActorTagが便利
- ActorTaggerユーティリティ
 - /Game/PCG/Utilities/ActorTagger/EUW_ActorTagger



Tag



Details panel for PCG_DitchASMTarget1 (Instance)

- BrushComponent (BrushComponent0) Edit in C++
- PCGComponent (PCG Component) Edit in C++
- tag
- General Actor LOD Misc Physics
- Rendering Streaming All
- PCG
 - PCGComponent
 - Tags
 - Component Tags 1 Array elements
 - Index [0] None
 - Tags
 - Component Tags 0 Array elements
 - Actor
 - Advanced
 - Tags
 - Index [0] PCG_ASM_OVERRIDE

Outliner panel showing the scene hierarchy

Item Label	Type
Lighting	Folder
DirectionalLight	DirectionalL
EditorSkySphere2	StaticMesh/
ExponentialHeightFog	Exponential
PostProcessVolume_	PostProces:
SkyAtmosphere	SkyAtmospl
SkyLight	SkyLight
VolumetricCloud	VolumetricC
PCG	Folder
Exclusion	Folder
PCG_DitchASMTarç	PCGVolume
PCG_DitchASMTarç	PCGVolume
PCG_DitchASMTarç	PCGVolume
PCG_DitchASMTarç	PCGVolume
PCG_DitchASMTarç	PCGVolume
PCG_DitchExclude	PCGVolume
PCG_DitchASMTæ	PCGVolume
PCG_DitchExclude2	PCGVolume
PCG_DitchExtraRoc	PCGVolume
PCG_DitchSurroun	PCGVolume
Location	Folder
PCGAreaDescriptio	TextRender.
PCGAreaDescriptio	TextRender.

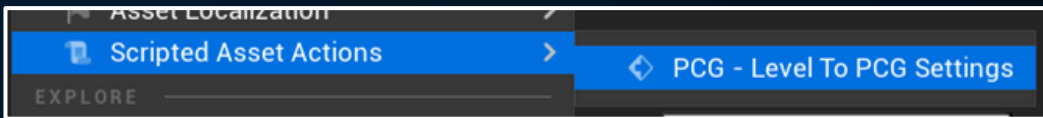
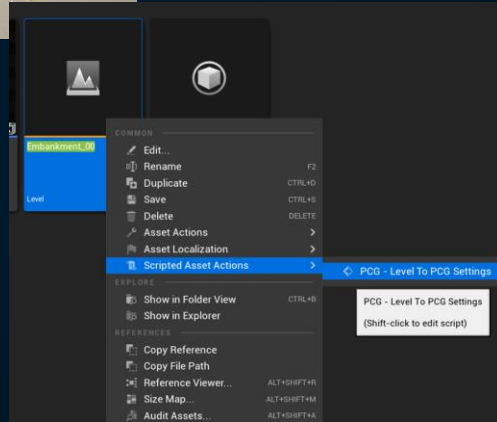
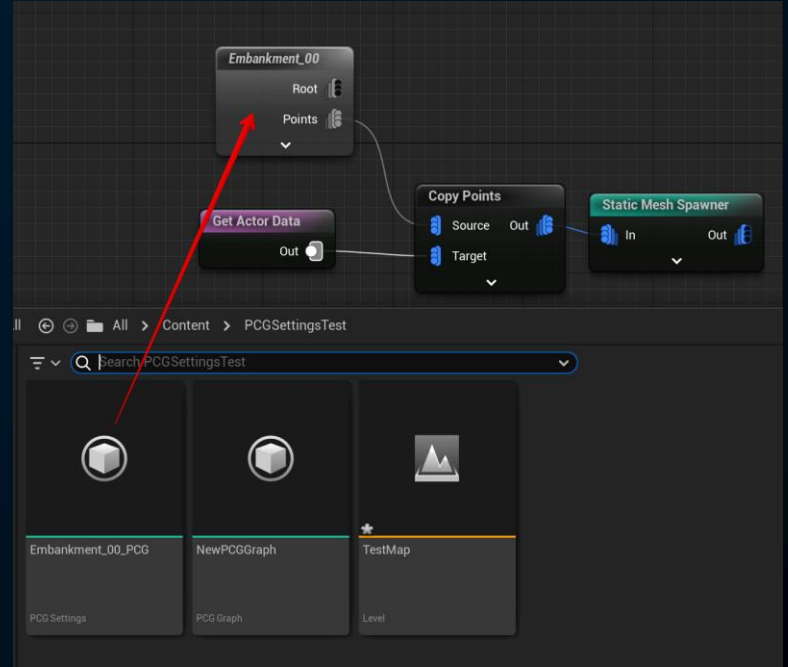
PCG Graph editor showing a Difference node and a PointsFromActorTag node.

The Difference node has Source and Differences inputs and an Out output. The PointsFromActorTag node has an Out output. The Actor Tag is set to PCG_ASM_OVERRIDE.

Settings for the Actor Tag:

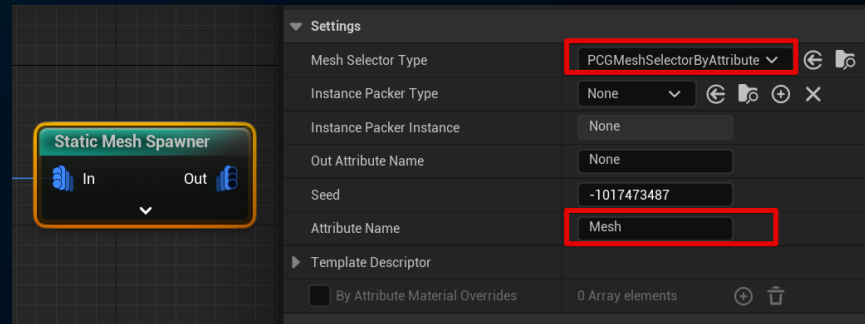
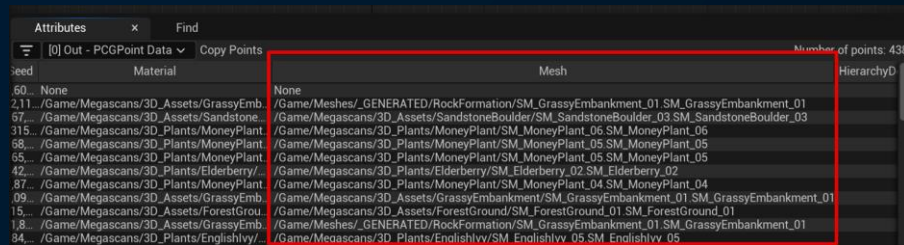
- Set bounds from:
- Apply Scale to B...:
- Bounds Min: -1.0 -1.0 -1.0
- Bounds Max: 1.0 1.0 1.0
- Tracked Actor T...: 0 Array elements
- Track Actors Onl...:
- Seed: -787091170

PCG Settings

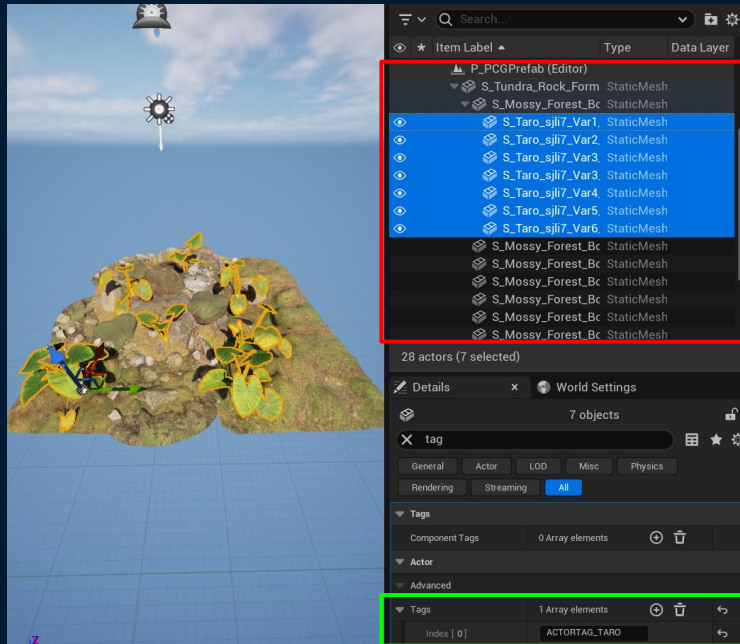


PCG Settings

- PCGGraphに直接インポートできるアセット
- ポイントデータを保持
- レベルから**PCGUtility_LevelToPCG**を使って作成
- PCGMeshSelectorByAttributeを使ってデータ中のメッシュ名で生成



PCG Settings



親子構造

ActorTag

The screenshot shows the PCG Graph editor. A graph is visible with nodes for 'P_PCGPrefab', 'Get Actor Data', and 'Copy Points'. The 'Copy Points' node has a table with columns for 'Hierarch', 'ActorInd', 'ParentId', and various 'Re' (Relevance) values. A red box highlights the 'Data' column, and a green box highlights the 'ACTORTAG_TARO' column. The text 'Number of points: 24' is visible in the top right corner of the table area.

Hierarch	ActorInd	ParentId	Re	Re	Re	Re	Re	Re	Re	Re	Re	ACTORTAG_BASEMENT	ACTORTAG_TARO
0	0	-1	0	0	0	0	0	1	1	1	1	0	0
1	8	0	0	0	0	0	0	0	0	0	0	0	0
2	13	22	0	0	0	0	0	0	0	0	0	0	0
2	14	22	0	1	...	0	0	0	0	0	0	0	0
2	15	22	50	...	0	0	0	0	0	0	0	0	0
2	16	22	...	0	0	0	0	0	0	0	0	0	0
2	17	22	...	10	0	0	0	0	0	0	0	0	0
2	18	22	...	2	0	0	0	0	0	0	0	0	0
2	19	22	...	20	0	0	0	0	0	0	0	0	0
2	20	22	...	80	10	0	0	0	0	0	0	0	0
2	21	22	...	90	20	0	0	0	0	0	0	0	0
1	22	0	0	0	0	0	0	0	0	0	1	0	0
2	23	22	...	0	0	0	0	0	0	0	2	2	0
2	24	22	...	0	0	0	0	0	0	0	2	2	0
2	25	22	...	0	0	0	0	0	0	0	1	0	0
2	26	22	...	2	0	0	0	0	0	0	2	2	0
2	27	22	...	0	0	0	0	0	0	0	1	2	2
3	28	13	...	1	...	0	0	0	0	0	1	0	1
3	29	13	...	3	...	0	0	0	0	0	1	0	1
3	30	13	...	2	...	0	0	0	0	0	1	0	1
3	31	13	...	3	10	0	0	0	0	0	1	0	1
3	32	13	...	3	...	0	0	0	0	0	1	0	1
3	33	13	...	6	...	0	0	0	0	0	1	0	1
3	34	13	...	6	...	0	0	0	0	0	1	0	1

参考リンク

- プロシージャル コンテンツ生成の概要 | 公式ドキュメント

<https://docs.unrealengine.com/5.2/ja/procedural-content-generation-overview/>

- Introduction to Procedural Generation plugin in UE5.2 | EDC

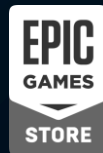
<https://dev.epicgames.com/community/learning/tutorials/j4xJ/unreal-engine-introduction-to-procedural-generation-plugin-in-ue5-2>

- [UE5] PCGの特徴と使い方 | 株式会社ヒストリア

<https://historia.co.jp/archives/34360/>

最後に

- メールでのお問い合わせ先
 - EGJ-support@epicgames.com
 - [カスタムライセンサー様向け]
 - 技術的課題や問題点に関するご相談
 - パフォーマンスプロファイリング・チューニング
 - 弊社製品に関するご契約などのご相談



- 新たにライセンスをご検討される方
 - <https://www.unrealengine.com/ja/license#contact-us-form>
- 過去の講演のスライド | Docswell
 - <https://www.docswell.com/user/EpicGamesJapan>
- Unreal Fest 2023 Tokyo プレイリスト
 - https://www.youtube.com/playlist?list=PLr_Cbd4sUDTzvdUTMe6cB5TVKDFZ8fMfC

