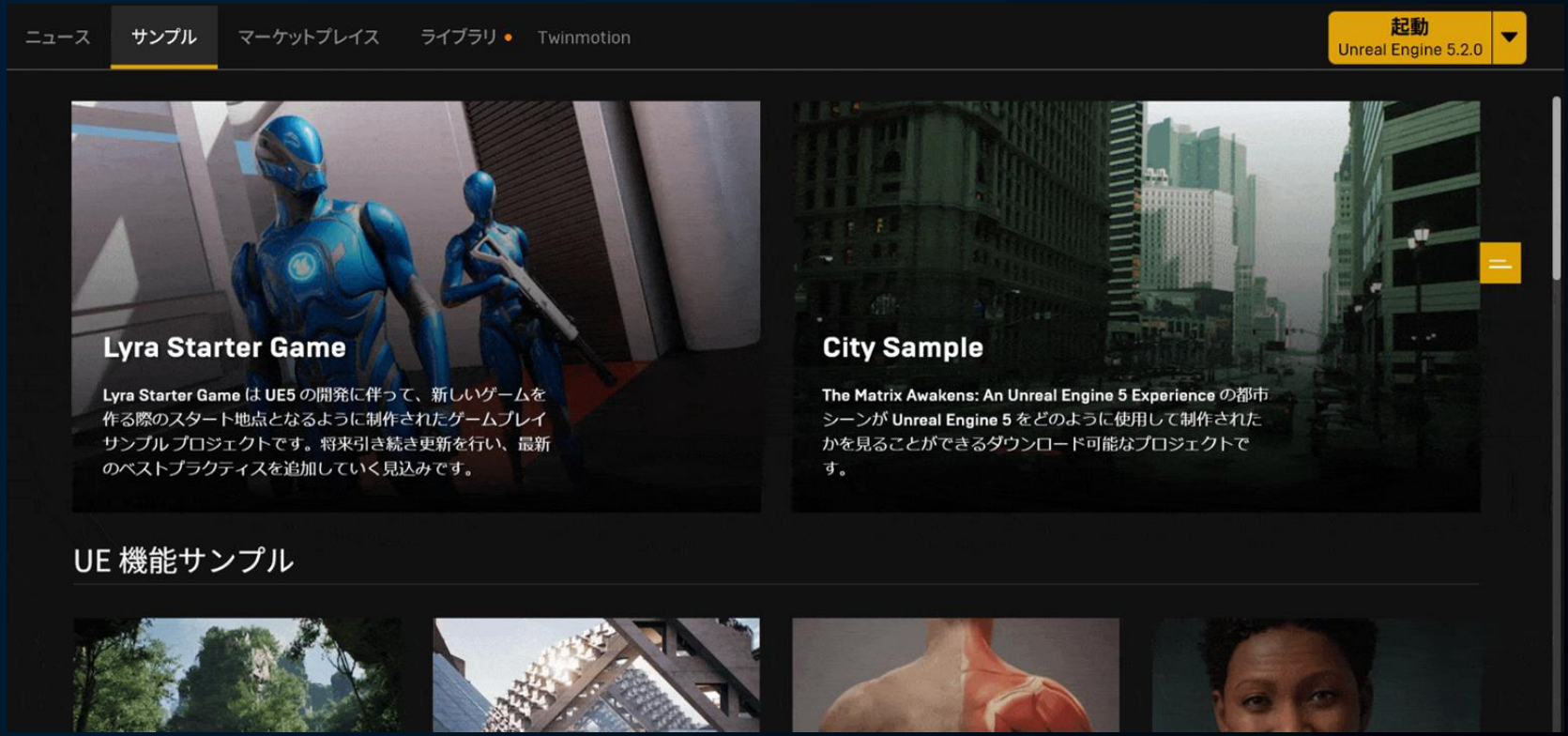
Unreal Engine 5.2 アップデート ～Rendering/PCG～

Epic Games Japan

Electric Dreams - 環境サンプル

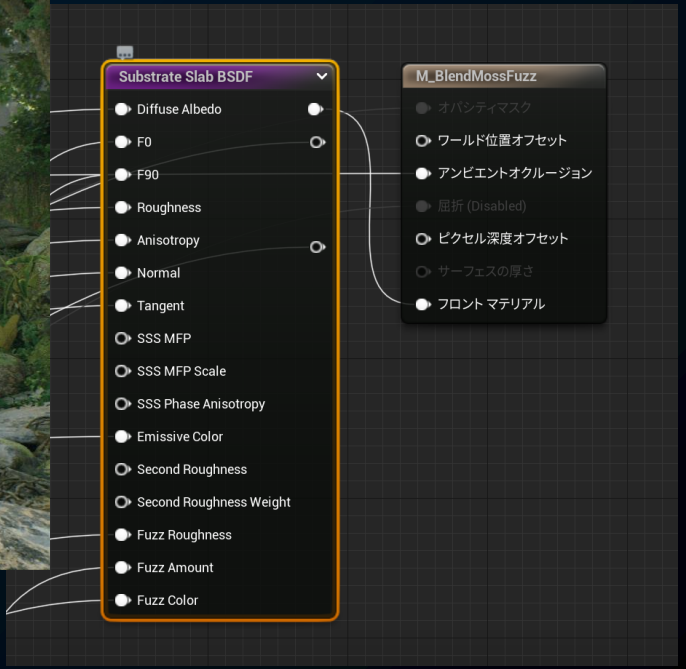


Electric Dreams - UE5.2

- **Substrate**
  - 新マテ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

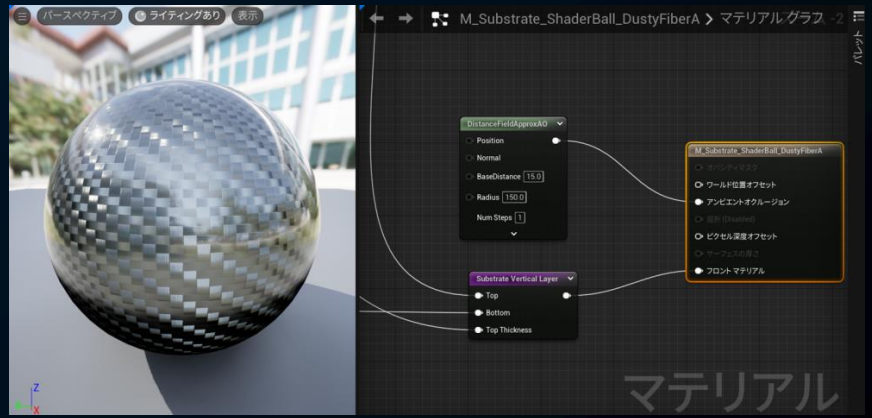


Substrate - 目的

・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

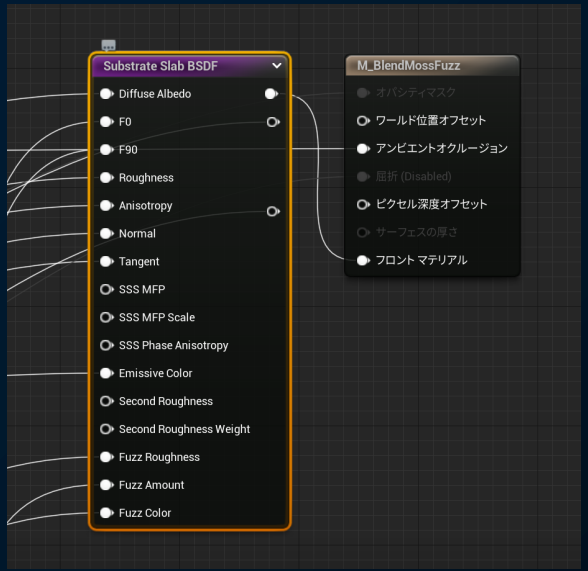
・表現力の向上を目指す
　→より複雑な混合、レイヤード表現をリアルタイムで

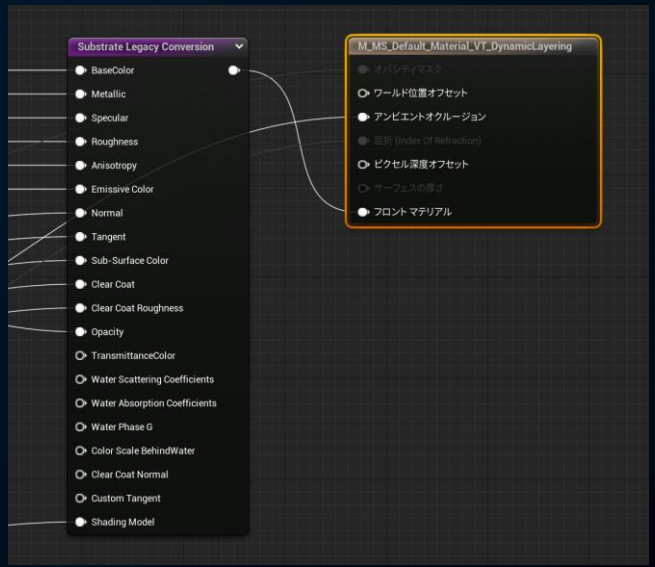
・互換性とパフォーマンス
　→従来と同様の使い方であれば
　　同程度のコストで表現できるように



Substrate - Electric Dreams 環境サンプル

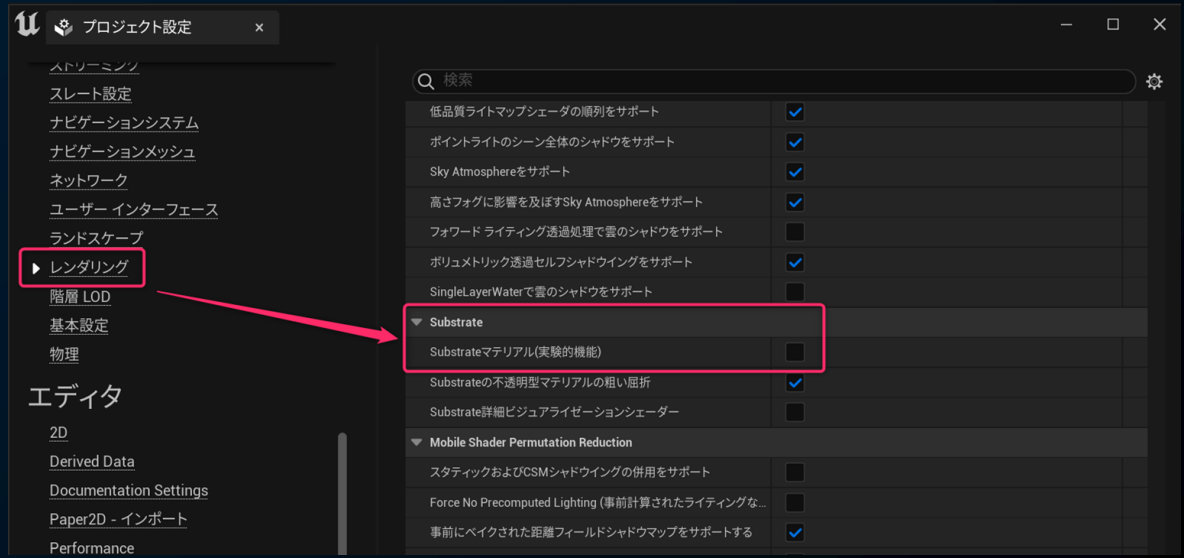
・Electric Dreamでも使用されているが...
従来のマテリアルから変換したマテリアルも多く存在
→ 複雑なものが多いため、初めて見る場合は**機能別サンプル**をおすすめ





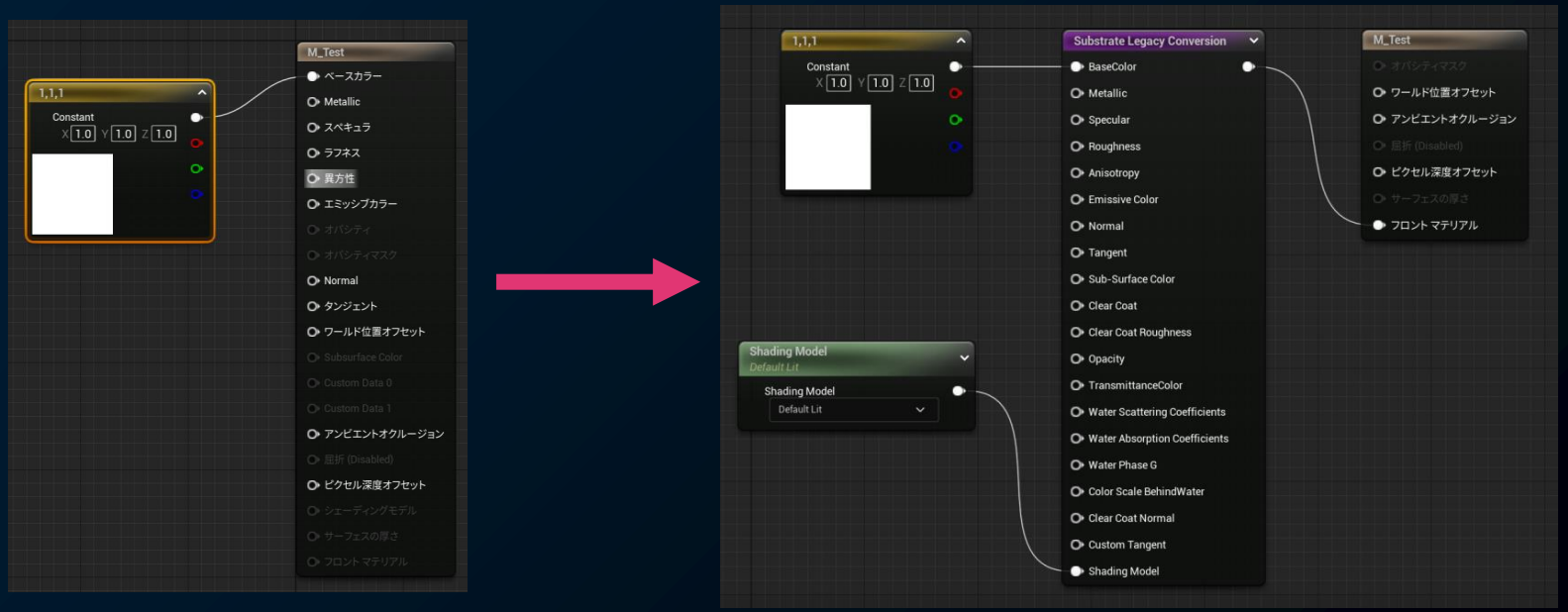
Substrate - 機能別サンプル

・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



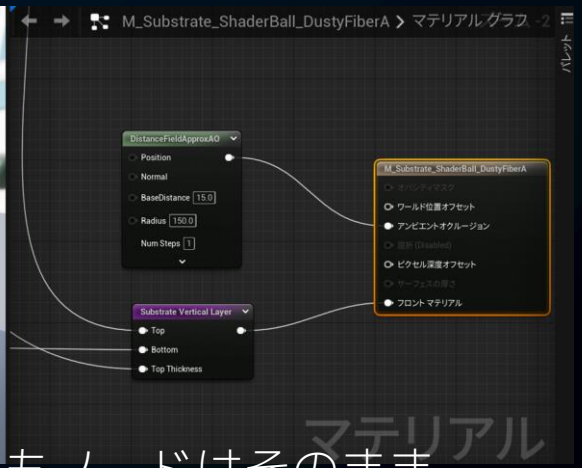
Substrate - 注意点

・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

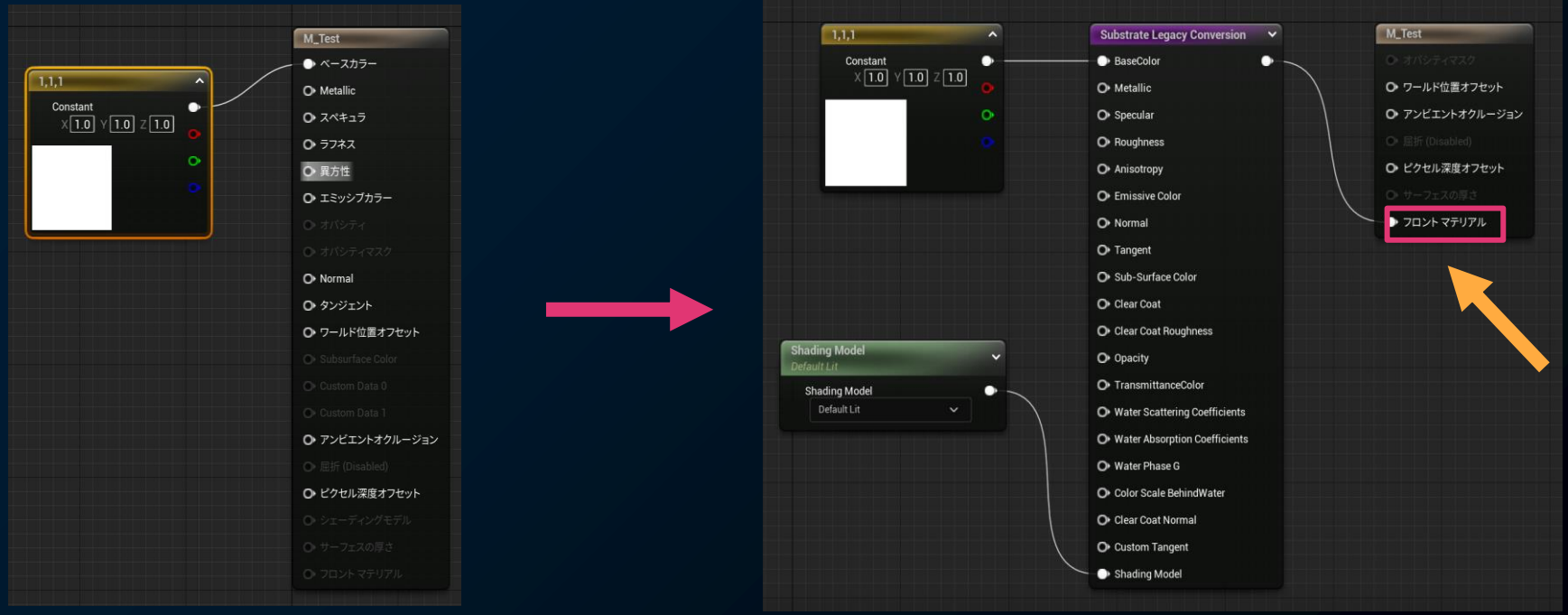
・変換した既存マテリアル保存してしまうと、
Substrate機能をプロジェクトから無効化しても**自動でマテリアルは戻らない**



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

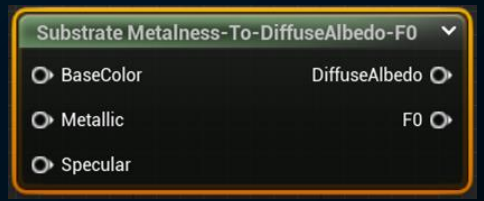
・これまでのマテリアルで設定していたシェーディングモデルはノードで指定
Substrate BSDF(双方向散乱分布関数)ノードとして用意されている

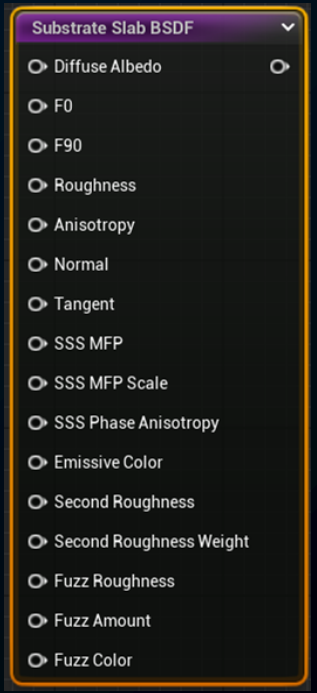
Substrate - Slab

・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

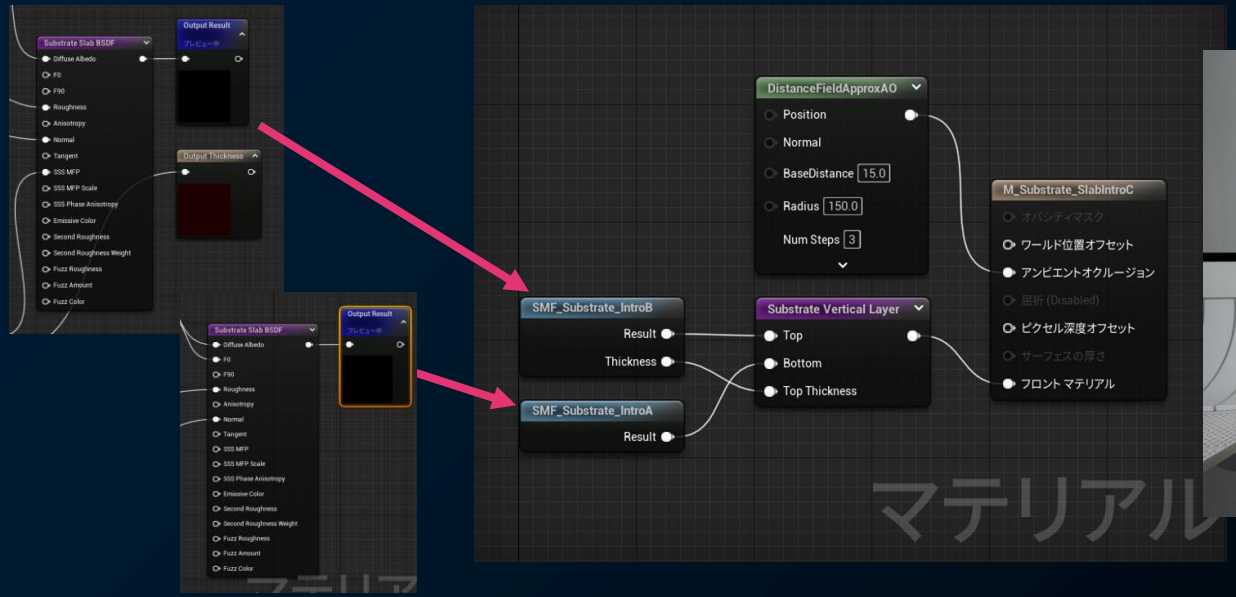


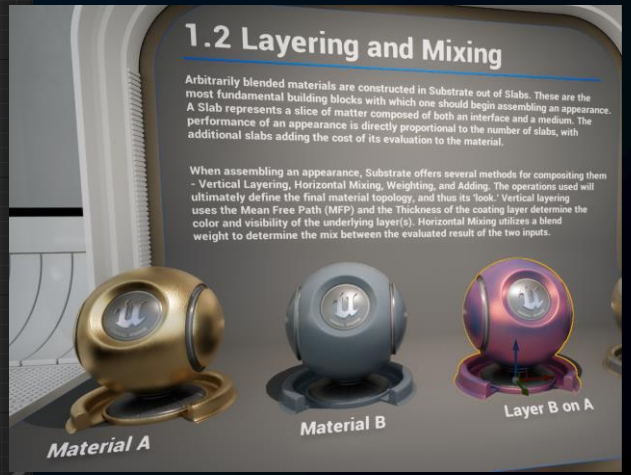


Substrate - Material Nodes - Substrate Materials
https://dev.epicgames.com/community/learning/courses/92D/unreal-engine-substrate-materials/8rXW/unreal-engine-substrate-the-slab

Substrate - Substrate Vertical Layer

・機能別サンプルの例 - Layer B on A マテリアル
Slub BSDF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

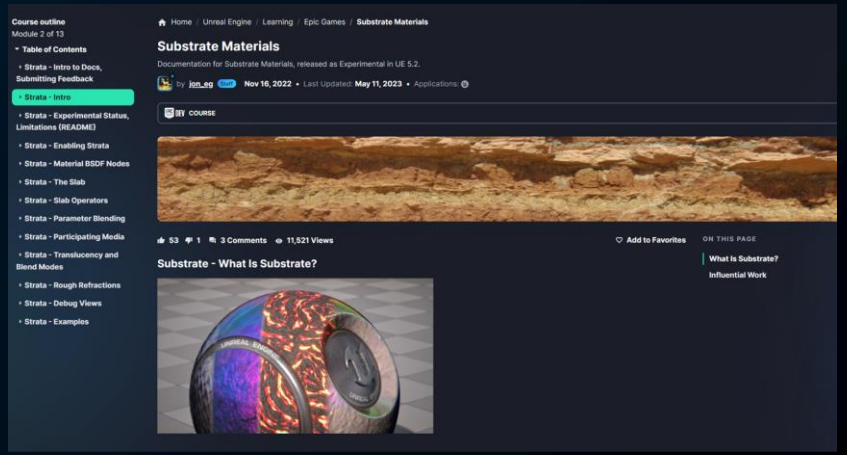




Substrate - ドキュメントとFeedback Thread

・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



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)

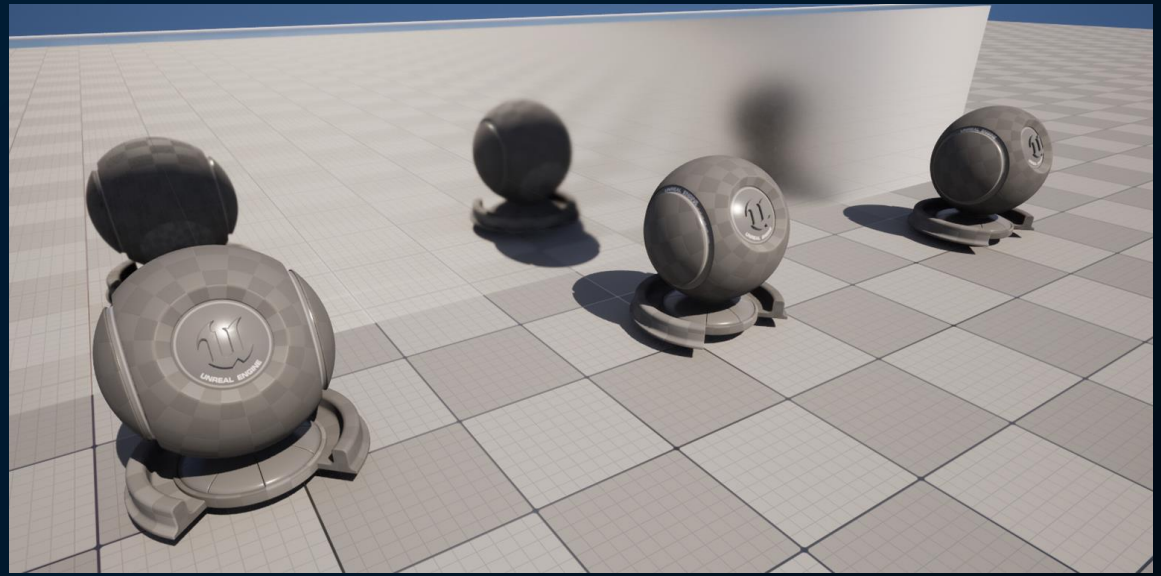
Lumen - GIやオクルージョンの品質向上

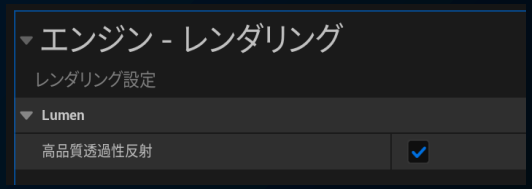
・薄いジオメトリに対する GI とオクルージョンが向上し、
Hair Groomとの統合も改善

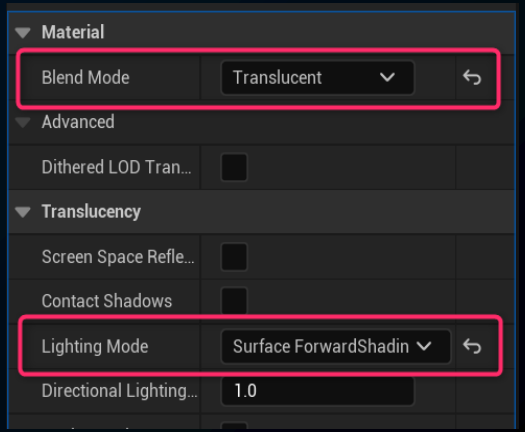
5.2リリースノートより

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)

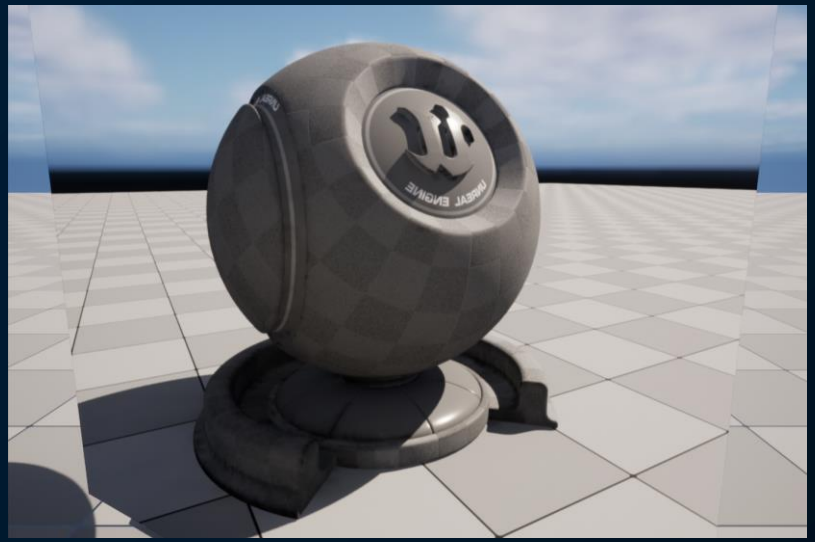




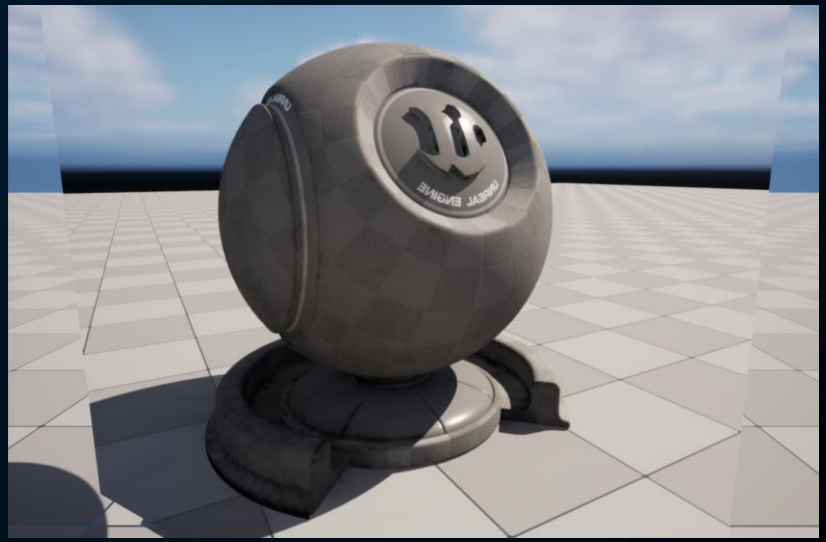


Lumen - リフレクションの品質向上(HWRT)

・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



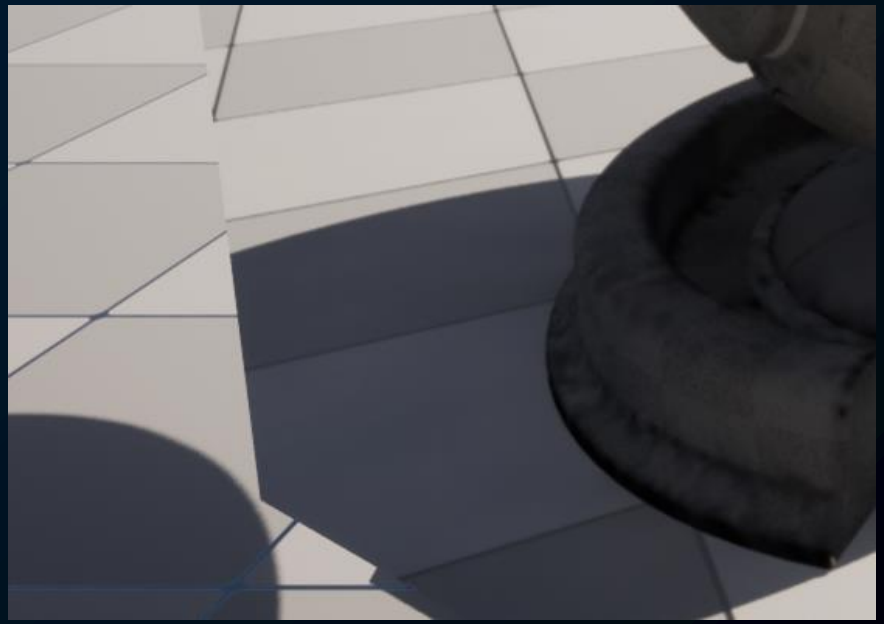
UE5.1



UE5.2

Lumen - リフレクションの品質向上(HWRT)

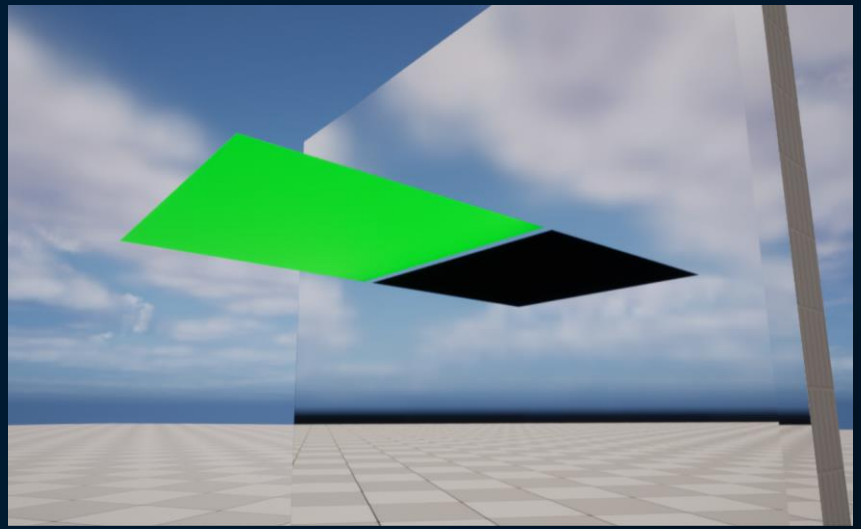
UE5.1

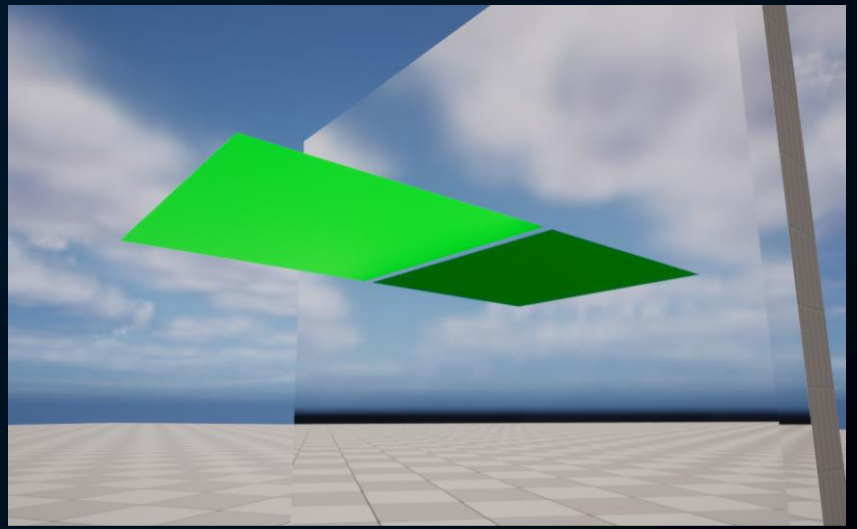


UE5.2

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



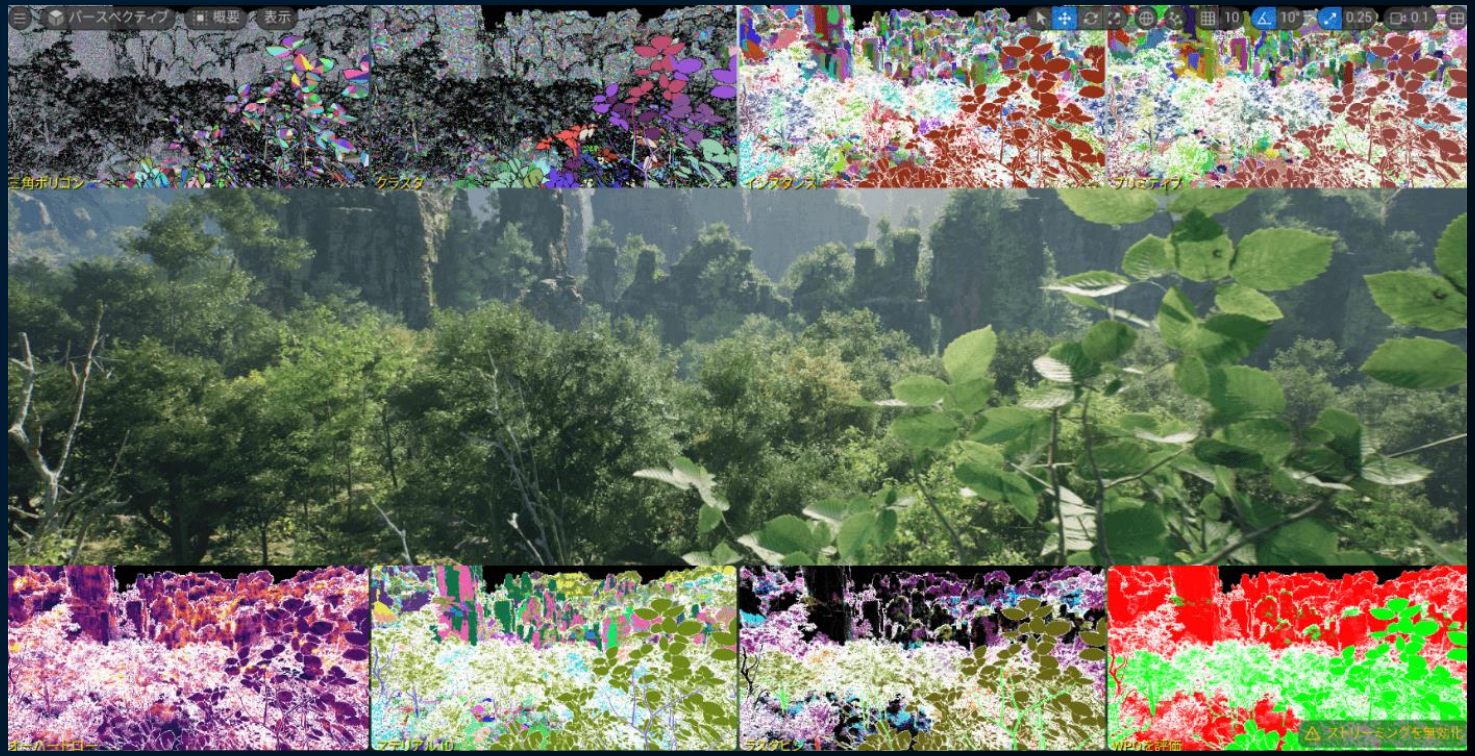


Lumen - パフォーマンスガイドの紹介

パフォーマンスを上げたい場合はこちらをチェック

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

Nanite

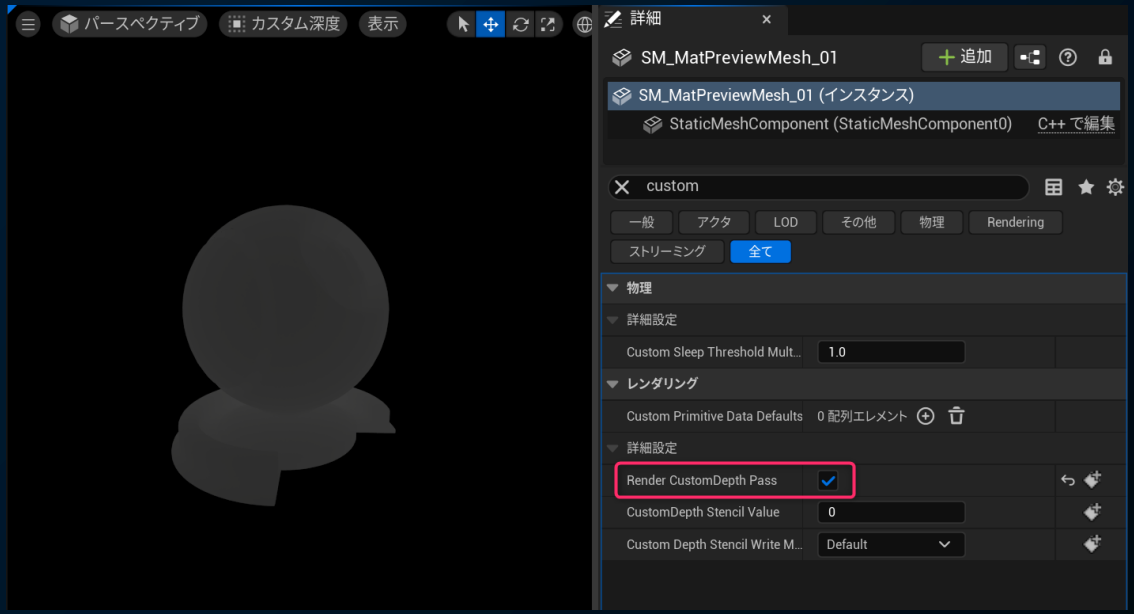

パースペクティブ
概要
表示
三角ポリゴン
クラスタ
インスタンス
プリミティブ
オーバードロー
マテリアルID
ランダムID
ストリーミングを無効化

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 ディスプレイスメントマッピング(実験的機能)

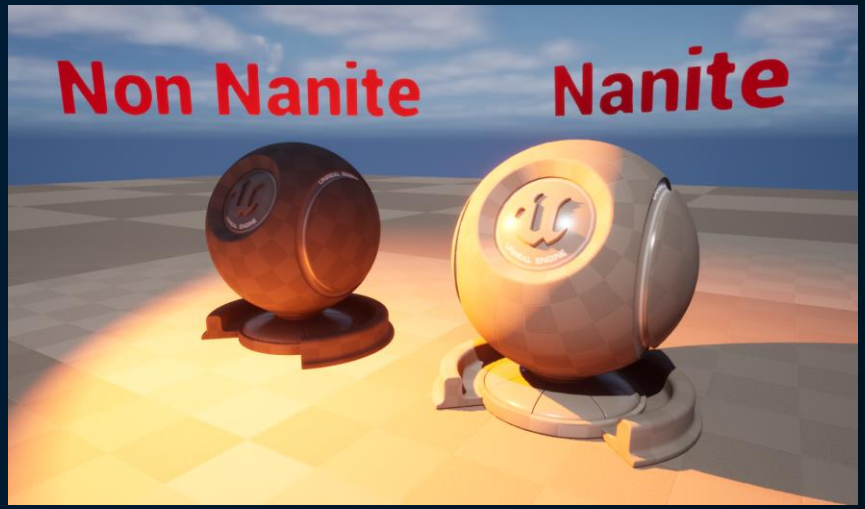
Nanite - Custom DepthとStencil対応

・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

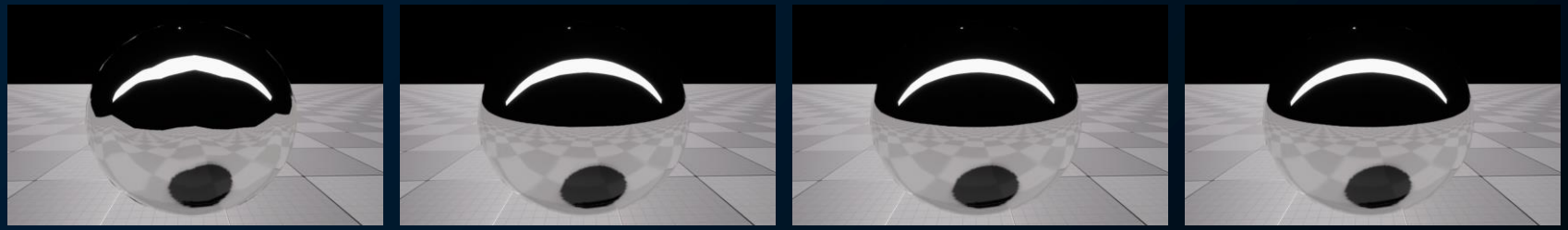
- **Planar Reflections**のNanite対応 (非Lumen環境)
  5.2でNaniteも反射を描画できるように

UE5.1

UE5.2

Nanite - 可変精度法線

Nanite - 可変精度法線



5 bits

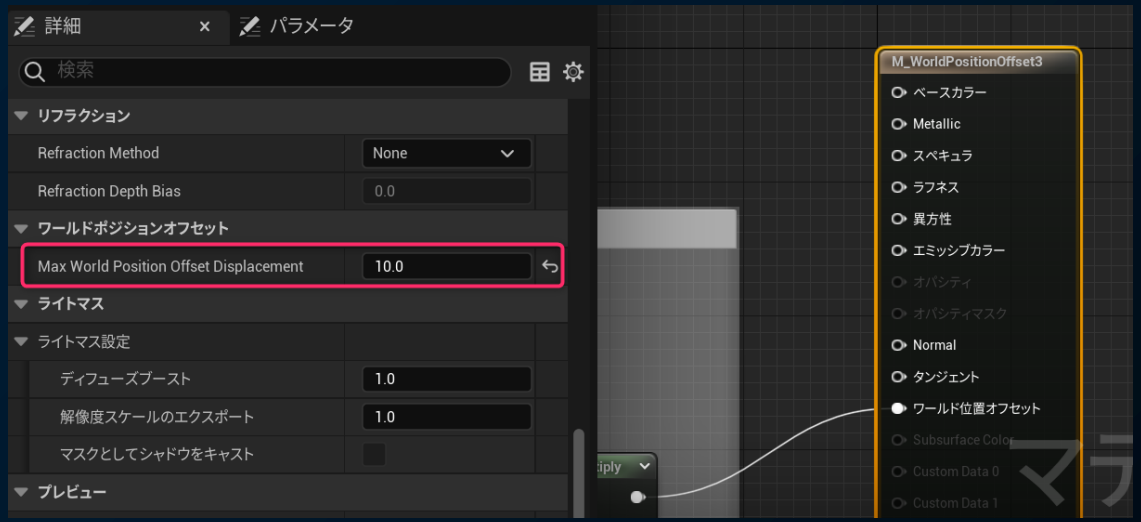
8 bits
デフォルト値

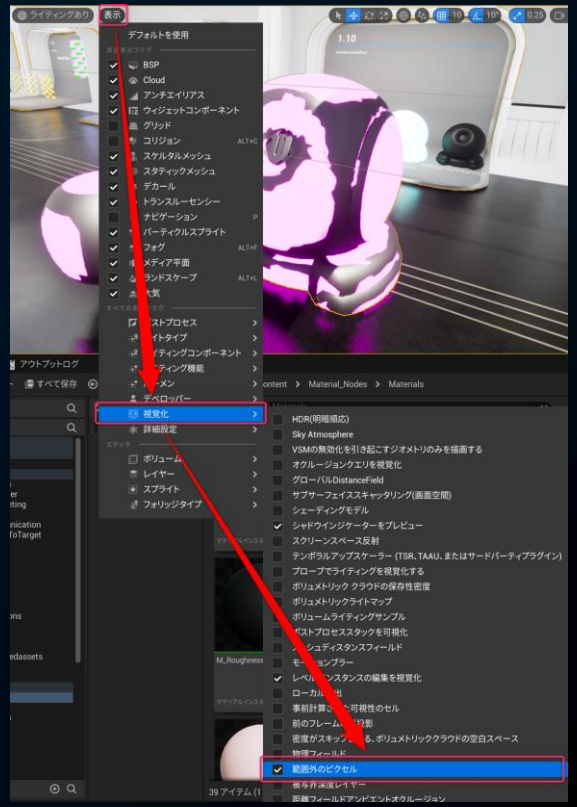
10 bits

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]

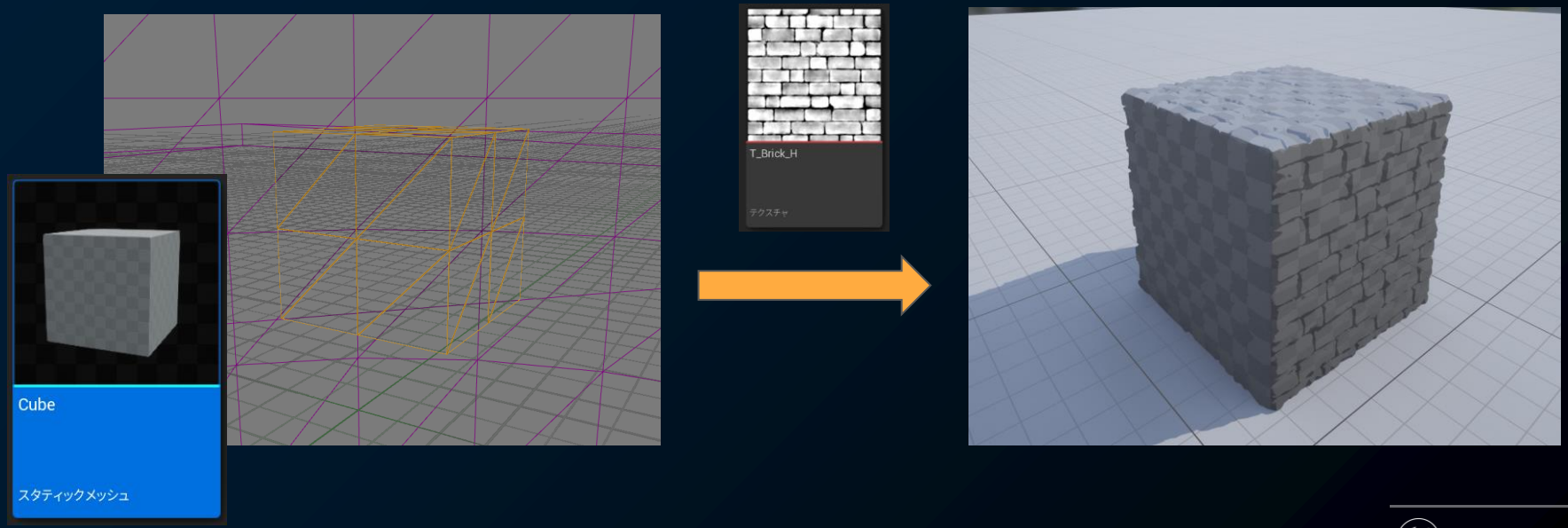
Nanite - Nanite Streamerの改善

・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***

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

・テクスチャと軽量なメッシュで、
　非破壊でディティールを追加した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

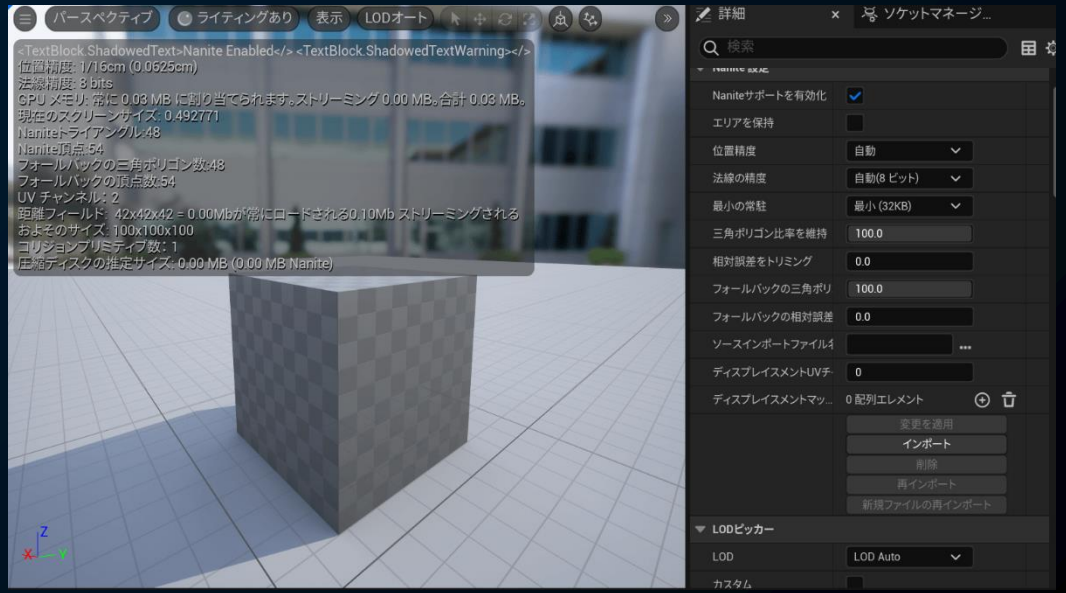
・**使用方法**
Nanite化したStatic Meshを開きNanite設定から設定を行う

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

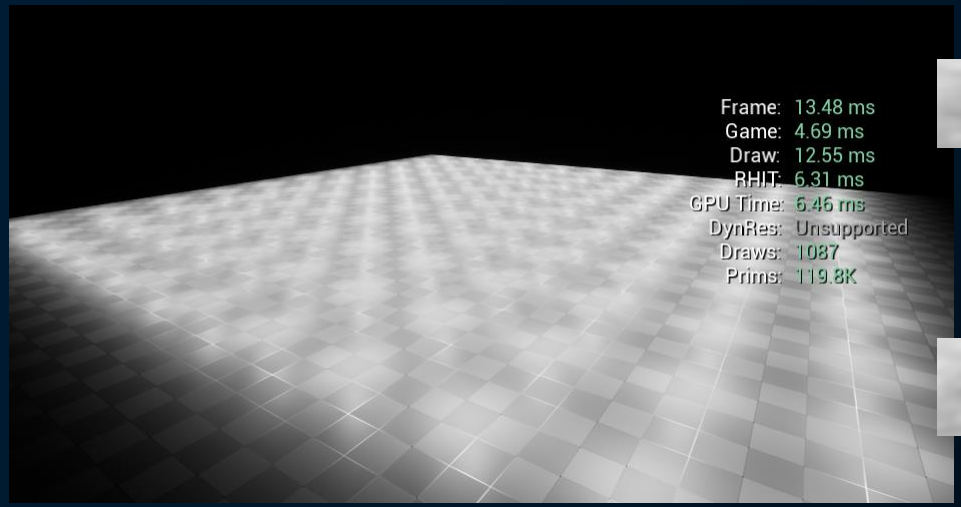
3. Magnitudeを調整
(押し出し量)

変更を適用で実行



VSM - シャドウに関する改善

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

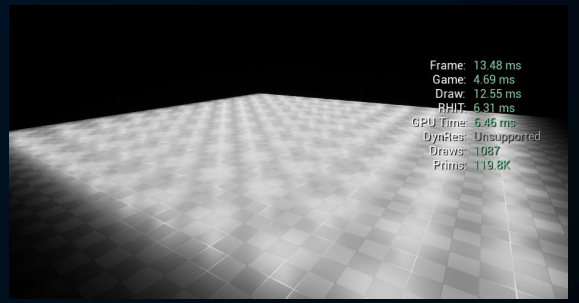


ポイントライト:100個配置
RTX4090にて計測

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

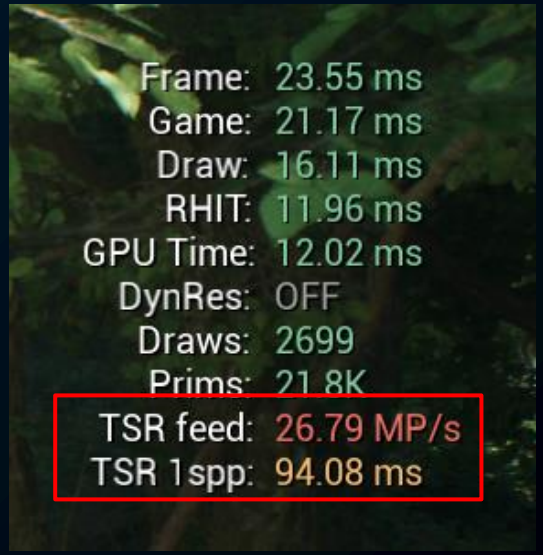
*= 描画解像度の幅 * 描画解像度の高さ * フレームレート*

※TSRに供給される情報量

TSR 1spp

*= 1000 / (スクリーン比率^2 * 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

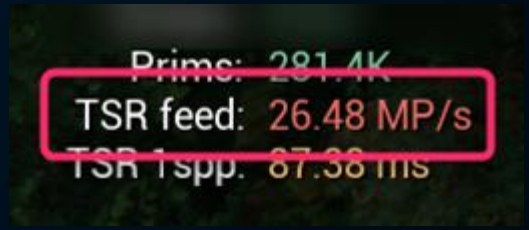


TSR - TSR Feed

・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

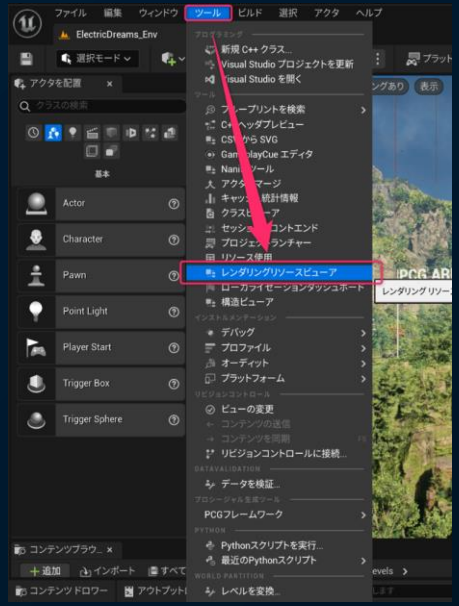
- TSRは過去フレームを蓄積させていく
  主にVelocity(速度)情報を元に判断 → Velocityがないと誤った蓄積が行われる → **残像**

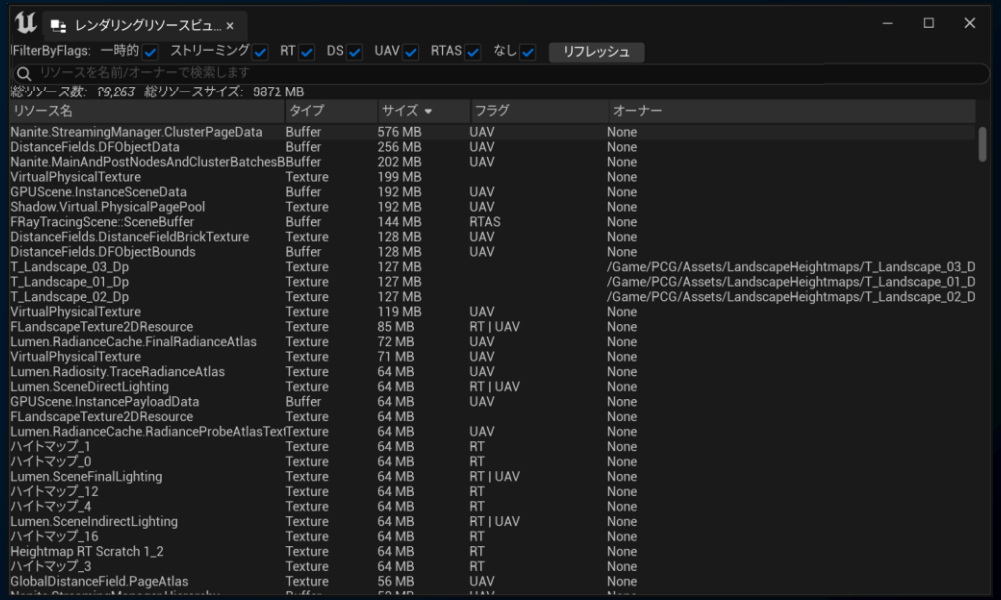
  ※対策(Velocityを書き出す)
  　各種マテリアルへの対応などを弊社篠山が記事にて公開

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

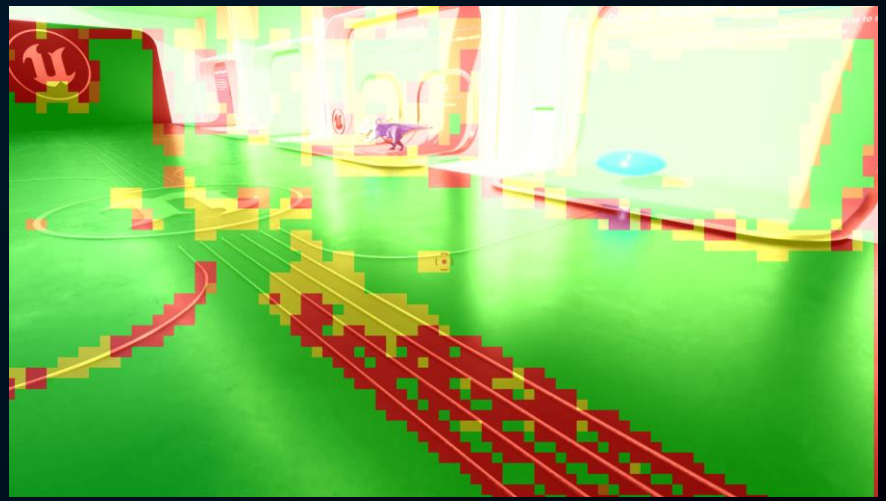
実験的機能

・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

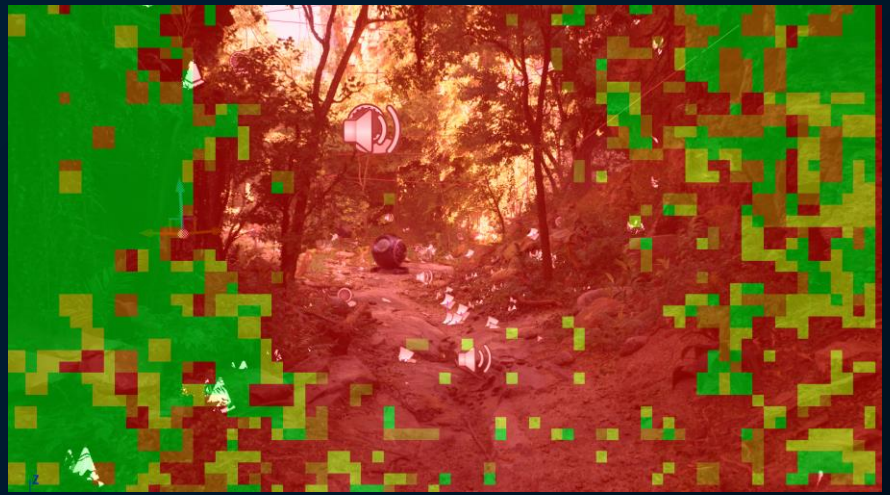
現時点ではパフォーマンス上、
劇的に改善することは確認できておらず、
実験的にお試し頂ければと思います。

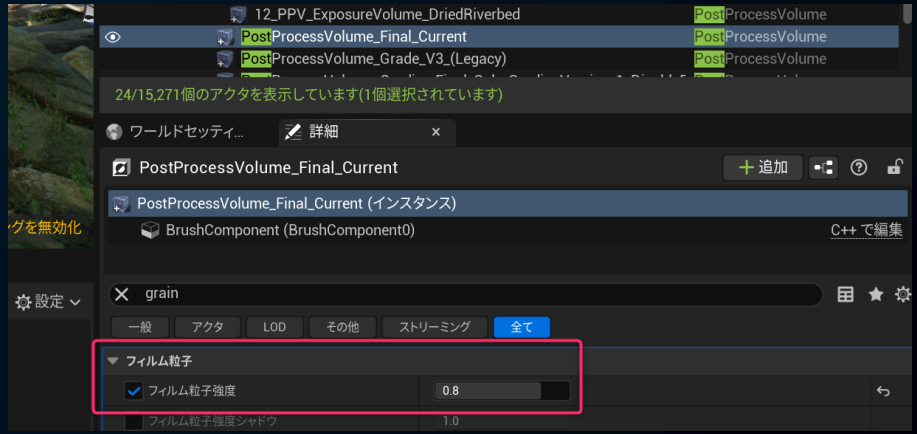


可変レート シェーディング

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





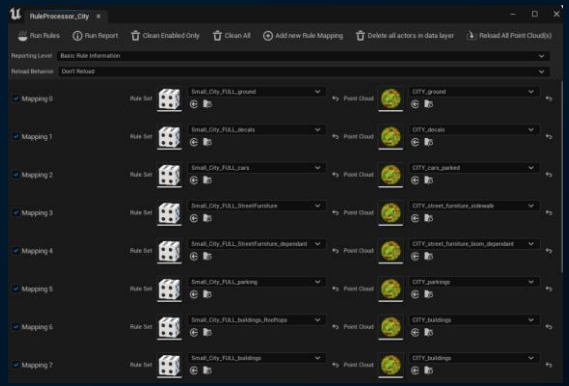
PCG

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 RuleProcesser



PCG フレームワーク

目的

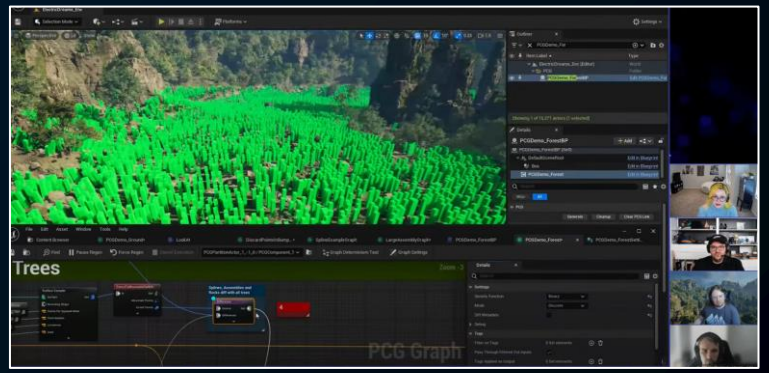
- **エンジン内蔵**
- プロシージャル生成
- 非破壊編集
- ノードベース & 拡張性

- 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=zlxpoY1TfkY

Procedural Tools In UE5 (日本語字幕)
https://www.youtube.com/watch?v=aoCGLW53fZg

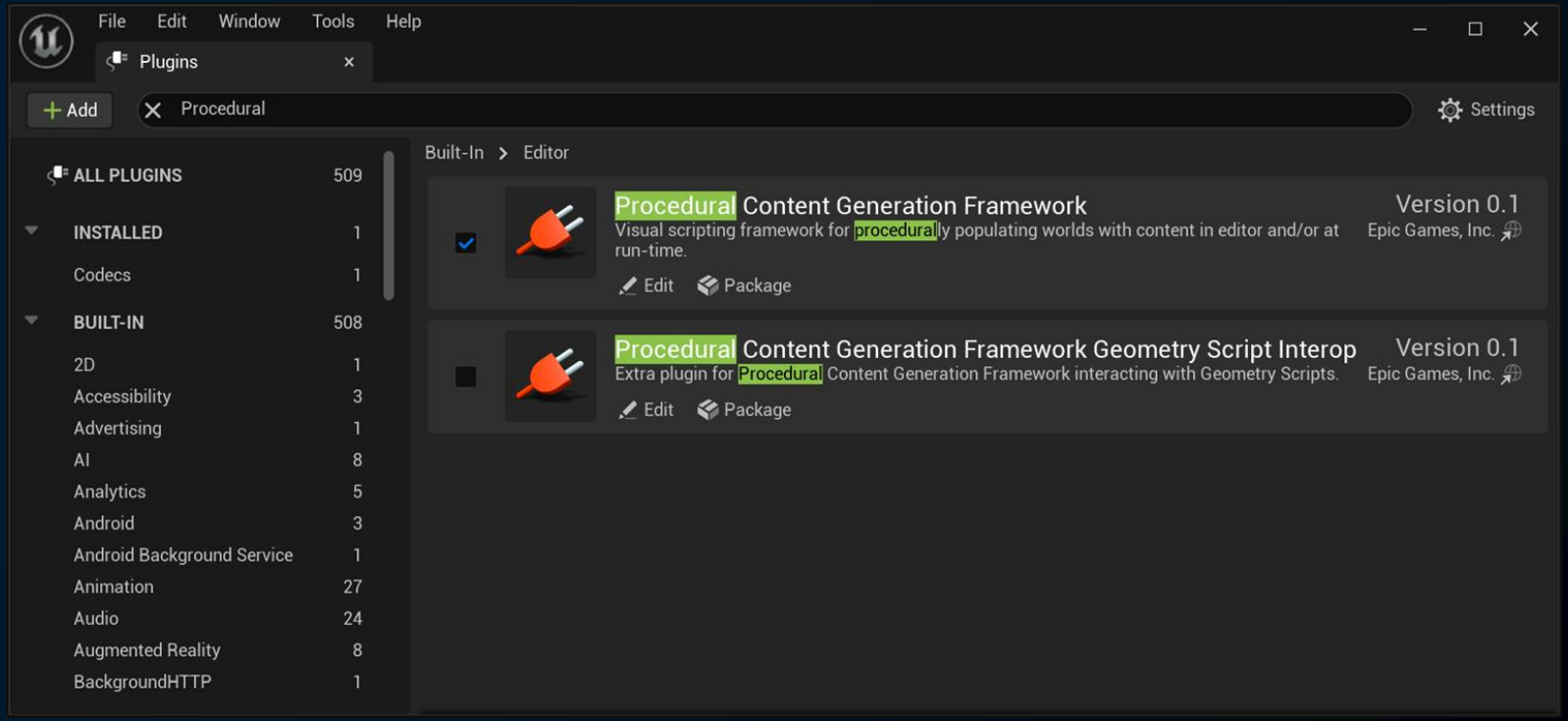


Deep Dive into the Electric Dreams Project | Inside Unreal
https://www.youtube.com/watch?v=6JUfisUhm68

Electric Dream environment サンプル

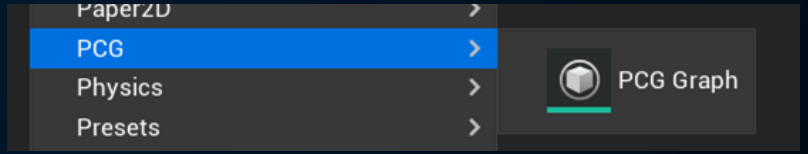
https://www.unrealengine.com/ja/electric-dreams-environment

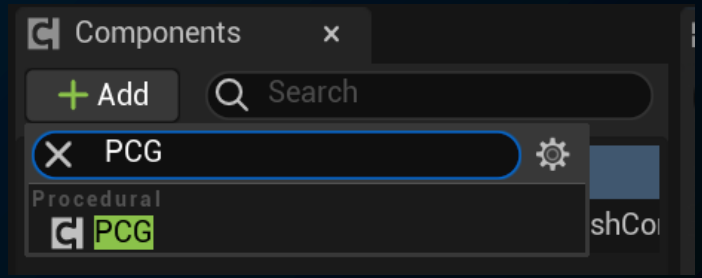
プラグインの有効化


File
Edit
Window
Tools
Help
Plugins
Add
Procedural
Settings
ALL PLUGINS 509
INSTALLED 1
Codecs 1
BUILT-IN 508
2D 1
Accessibility 3
Advertising 1
AI 8
Analytics 5
Android 3
Android Background Service 1
Animation 27
Audio 24
Augmented Reality 8
BackgroundHTTP 1
Built-In > Editor
Procedural Content Generation Framework
Visual scripting framework for procedurally populating worlds with content in editor and/or at run-time.
Version 0.1
Epic Games, Inc.
Edit
Package
Procedural Content Generation Framework Geometry Script Interop
Extra plugin for Procedural Content Generation Framework interacting with Geometry Scripts.
Version 0.1
Epic Games, Inc.
Edit
Package

PCG フレームワーク

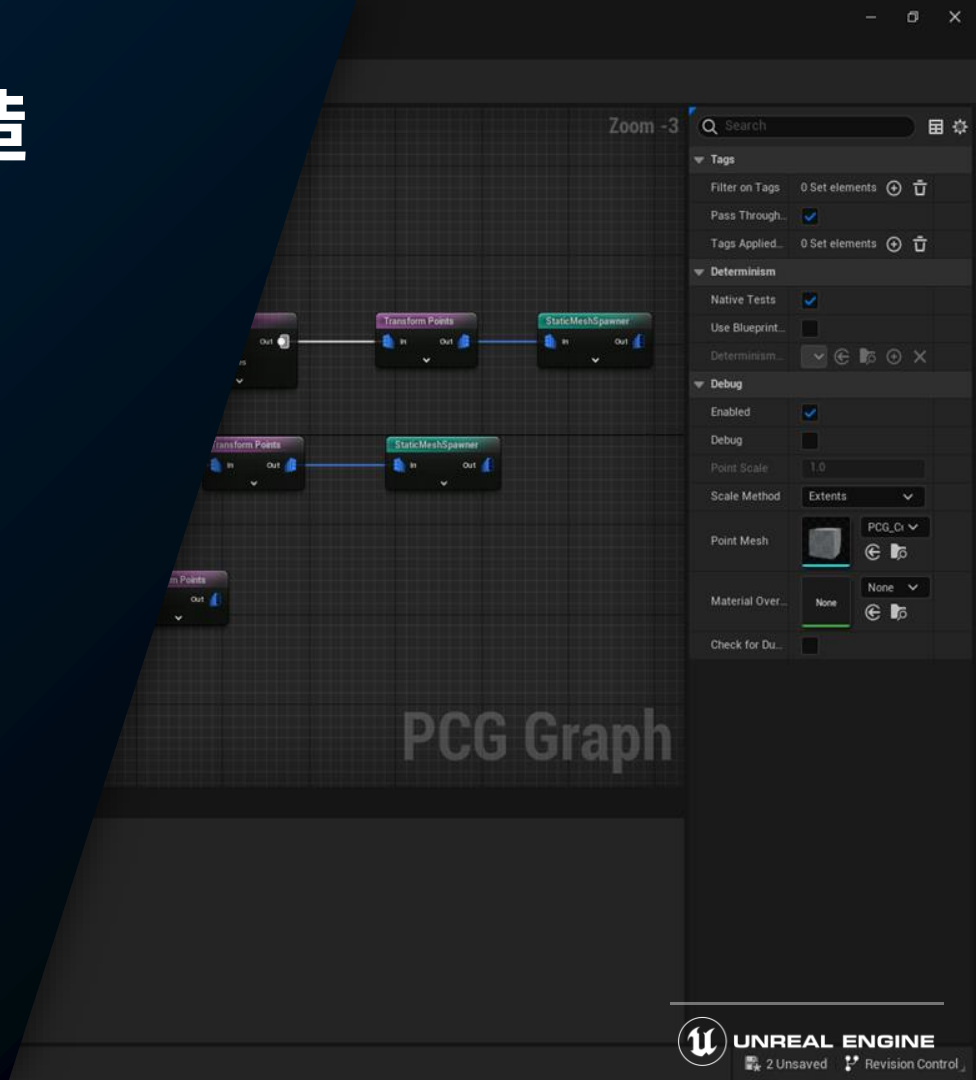
- PCGGraph
  - マテリアルのようなノードグラフ
- PCGComponent
  - PCGGraphを実行するコンポーネント
- PCGVolume
  - PCGComponentとボリュームを持ったアクター





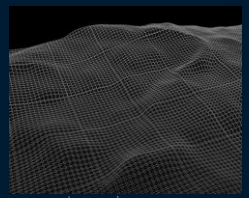
PCGGraphのノード構造

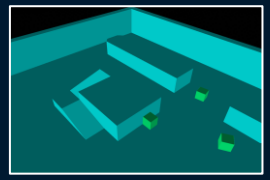
- データの取得
- ポイントの生成
- ポイントの加工
  - フィルタ
  - 変形
  - 演算
- 配置

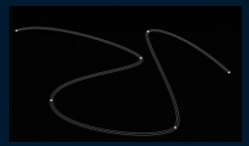


PCGGraphの構造

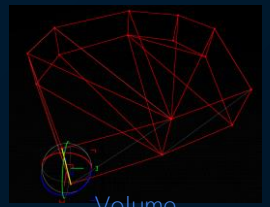
空間データ → ポイント生成 → 加工 → 配置


Landscape

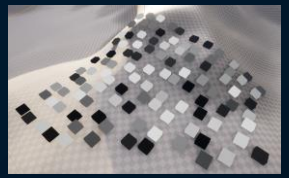

Collision


Spline

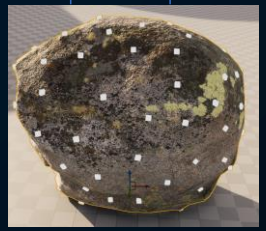

Texture

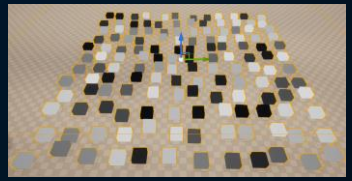

Volume

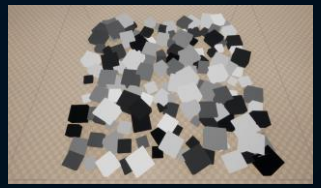

PCGSetting


SurfaceSampler


SplineSampler

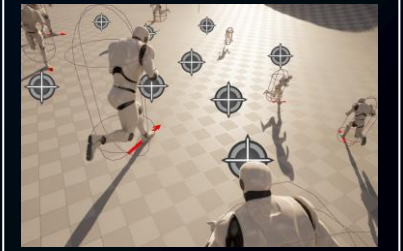

MeshSampler


Filter

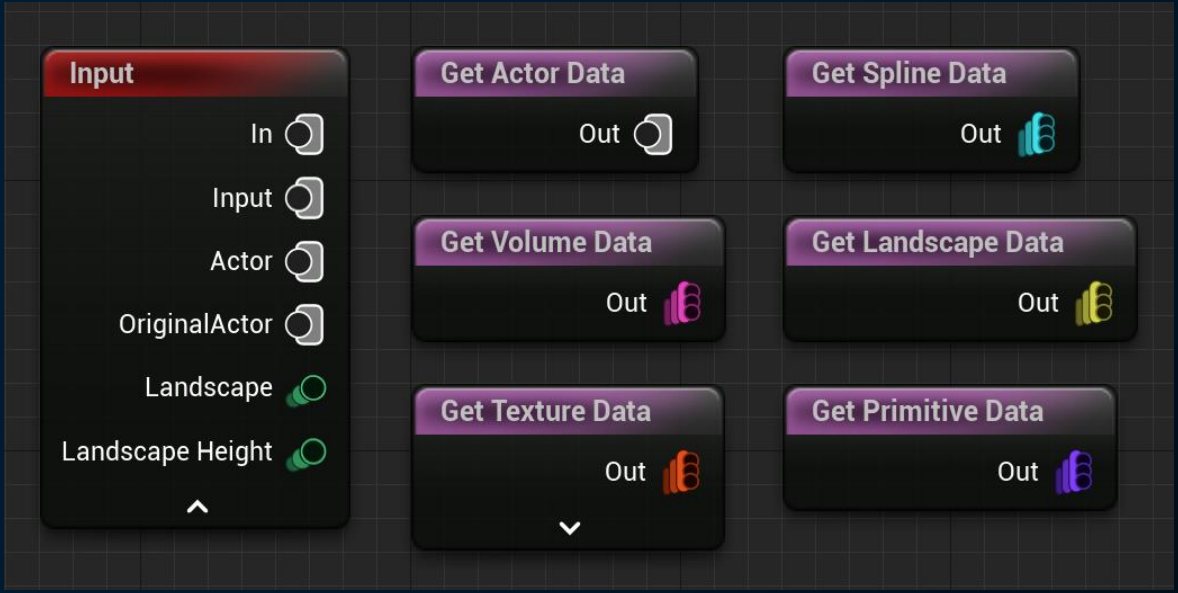

Pruning


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

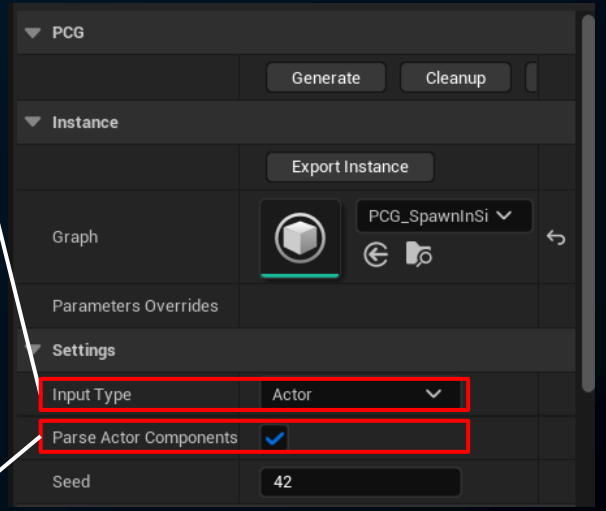
PCGコンポーネントのInputTypeにより変化
コンポーネントがパーティション化されている
→ パーティションアクター

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

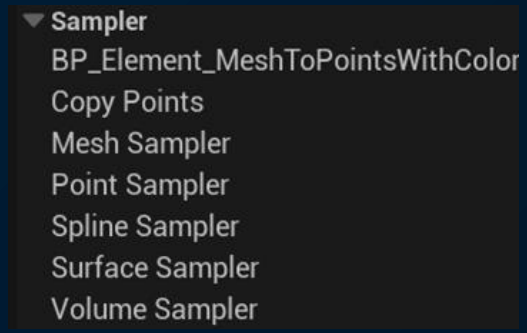
上記に当てはまらず ParseActorComponent:True
→全プリミティブをユニオンしたもの

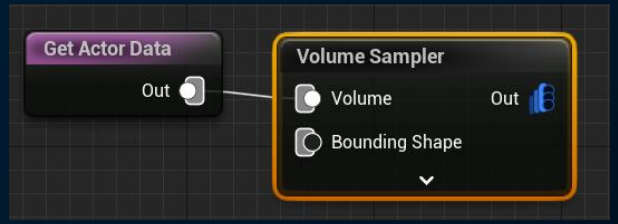
上記全てに当てはまらず ParseActorComponent:False
→アクターのバウンズ

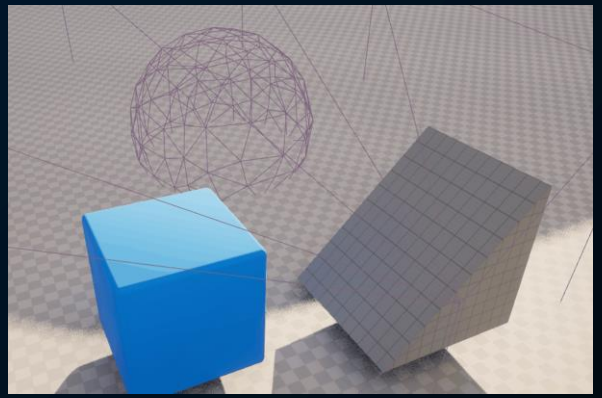


詳しくは
UPCGComponent::CreateActorPCGDataCollection
を参照してください

代表的なポイント生成ノード

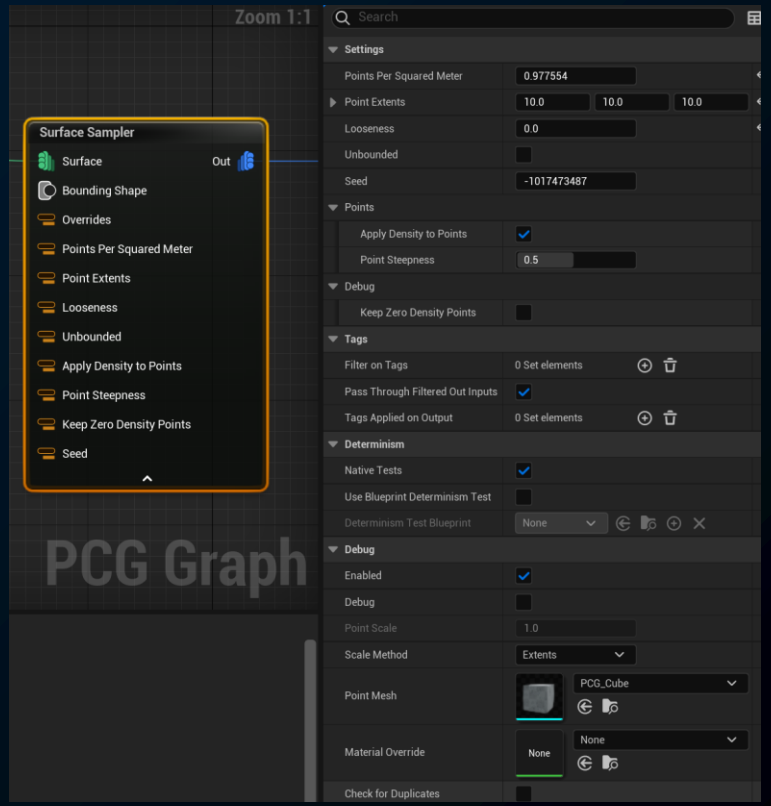




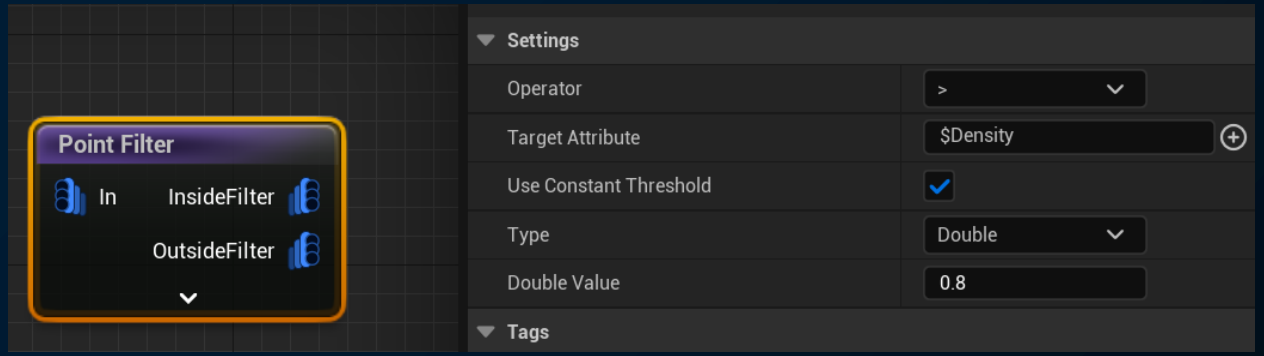


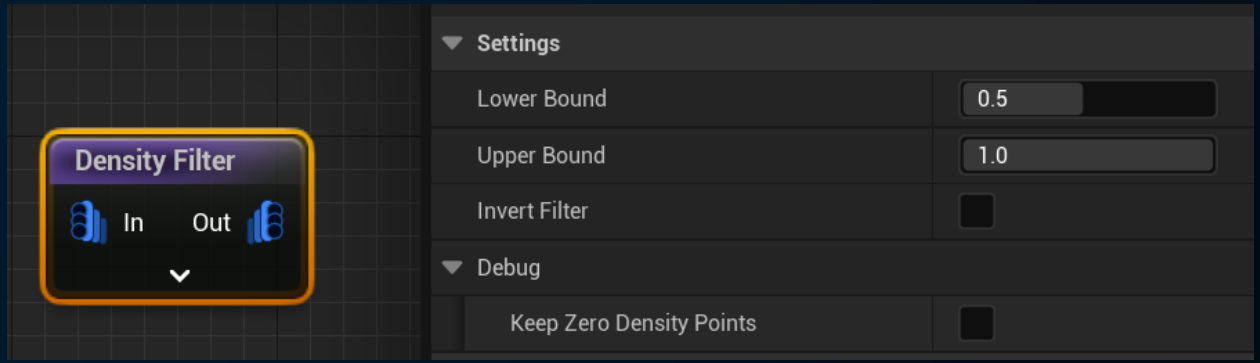
SurfaceSampler

- 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
  - バウンズ*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

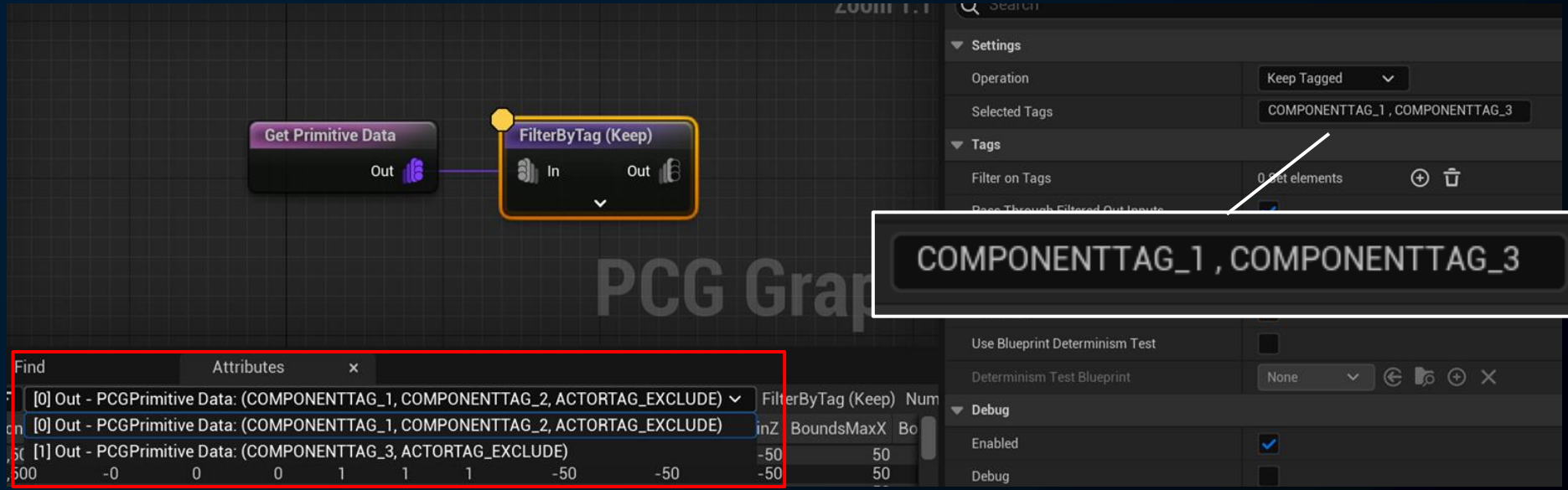


代表的な加工ノード (フィルタ1)

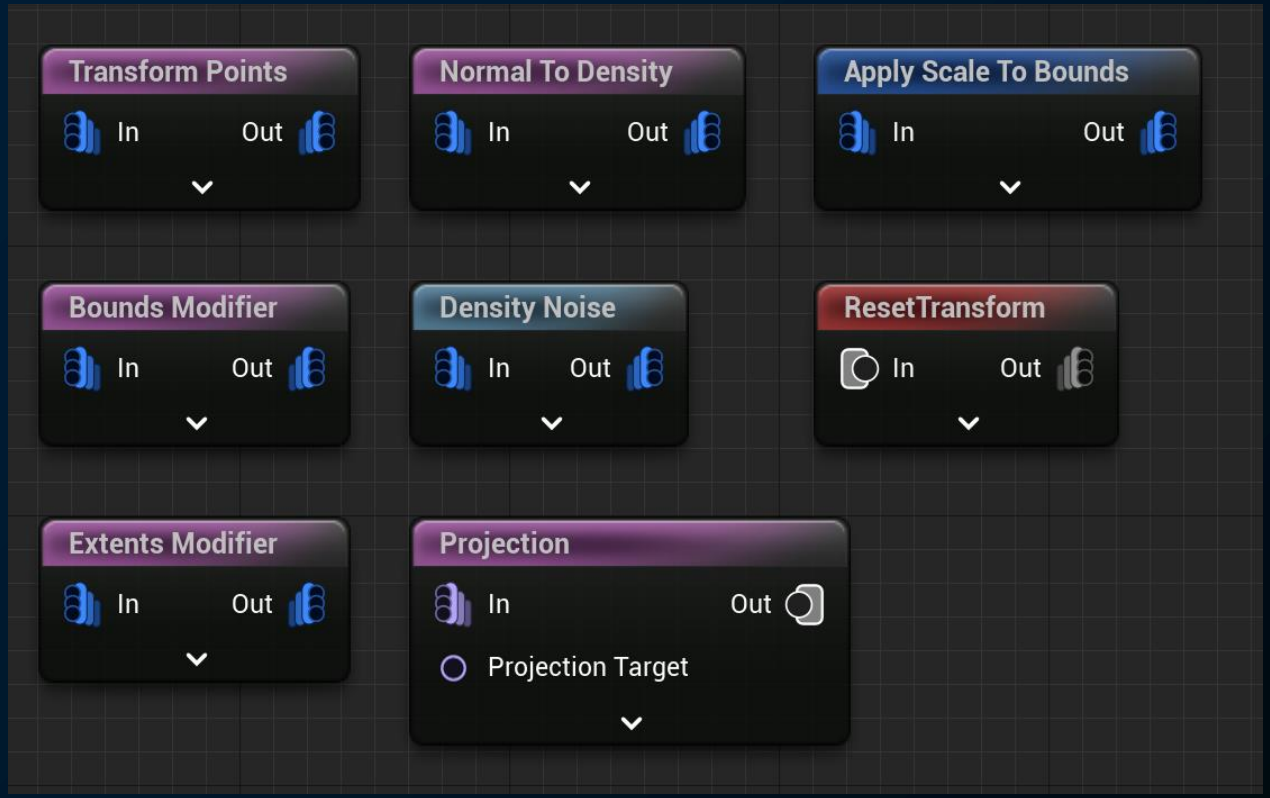




代表的な加工ノード (フィルタ2)



代表的な加工ノード（ポイント）

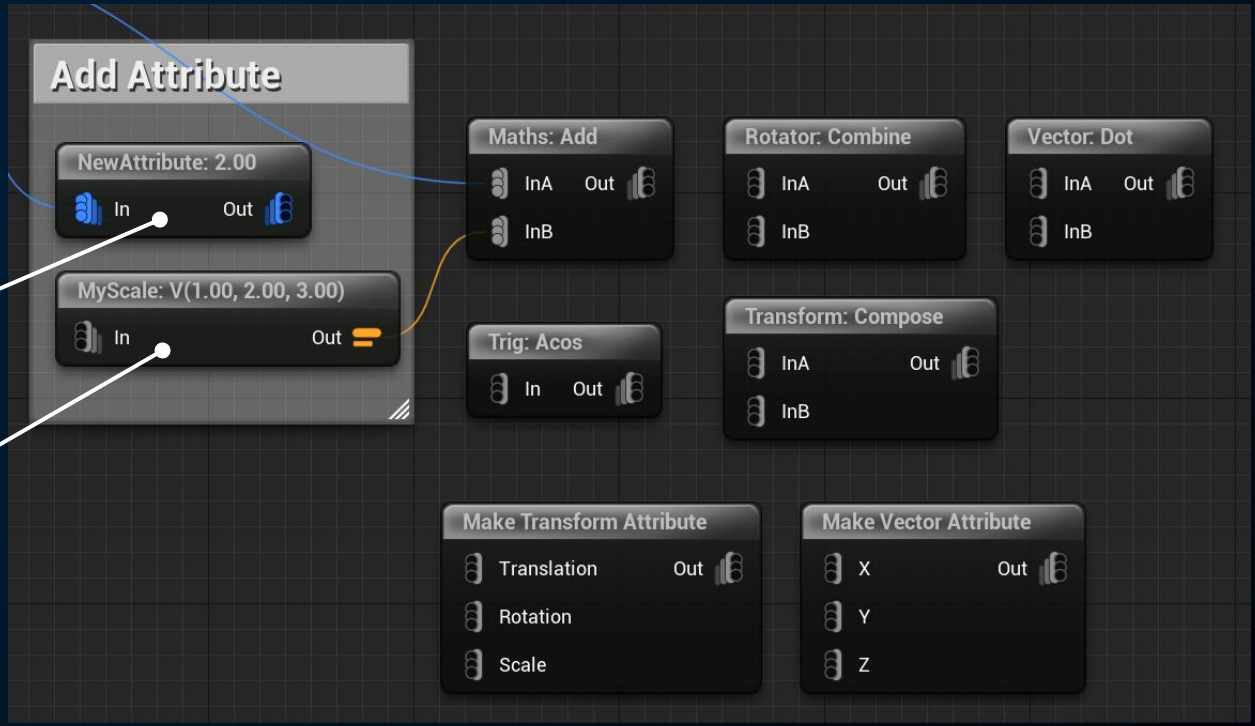


代表的な加工ノード（点群に対する演算）

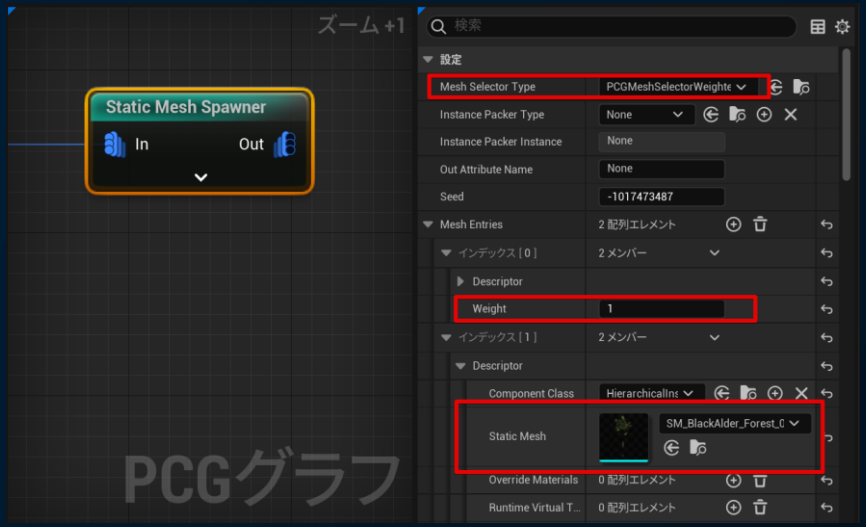
代表的な加工ノード（変数演算）

全てのポイントに
変数を追加

定数を作成



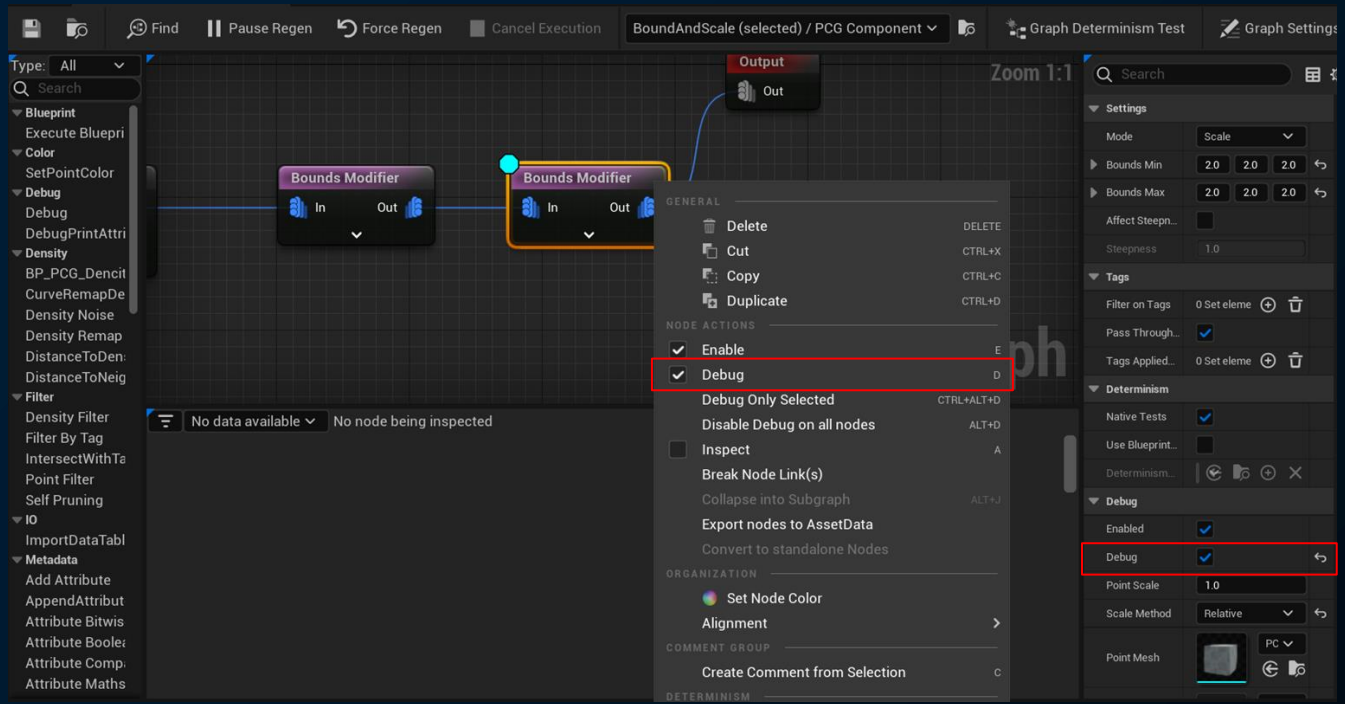
代表的な生成ノード

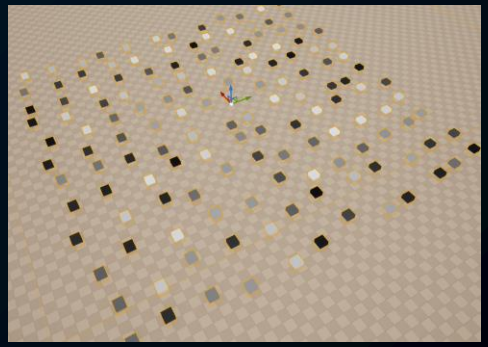


TIPS

- 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

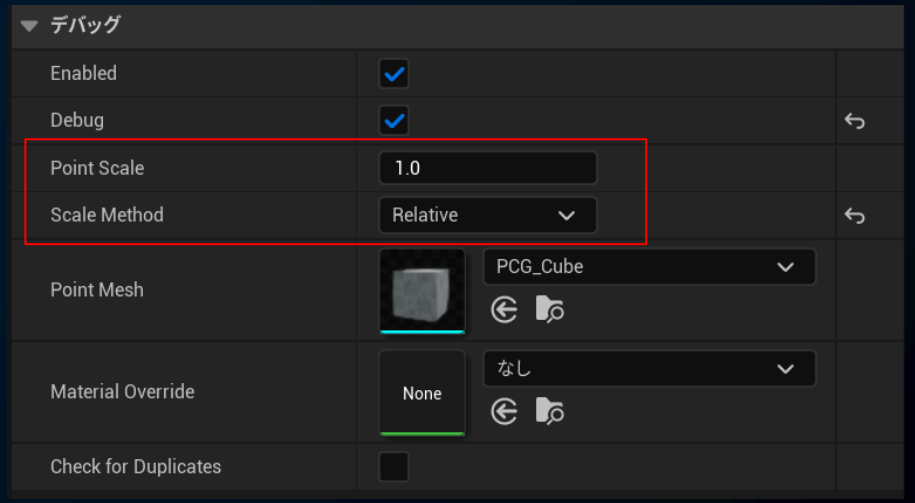




UNREAL ENGINE

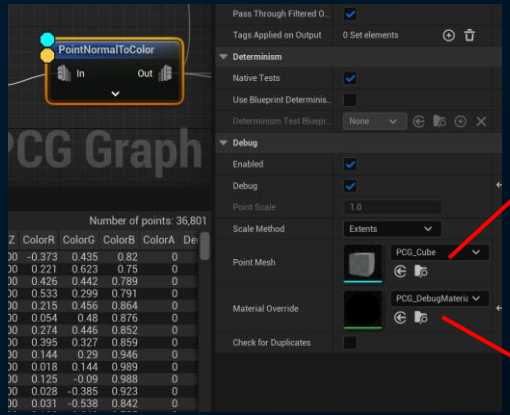
デバッグ

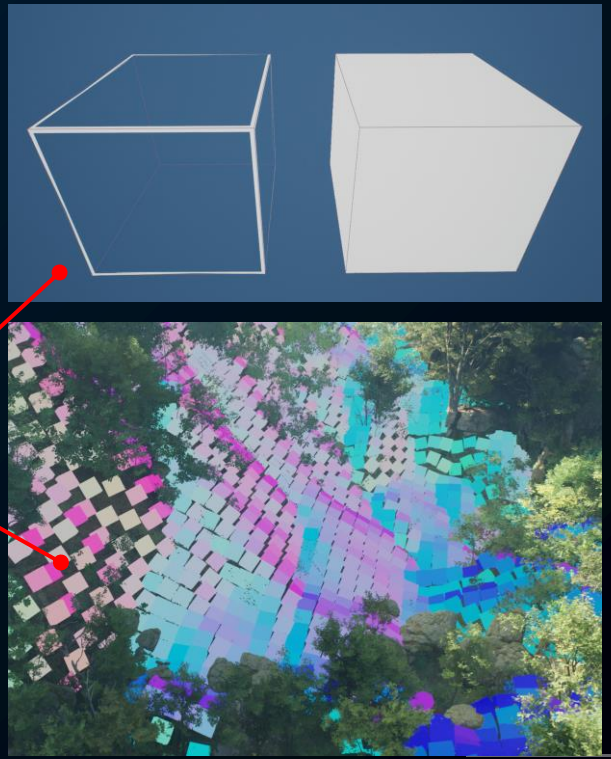
- ScaleMethod
  - Relative
    - **TransformのScale** * Point Scale
  - Absolute
    - Point Scale
  - Extents
    - **Boundsの大きさ**



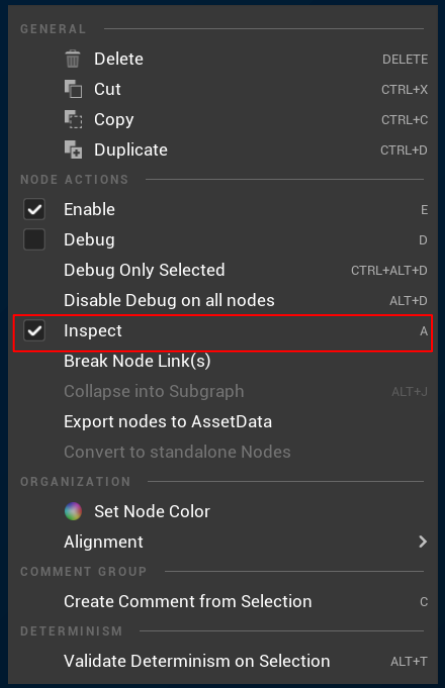
デバッグ

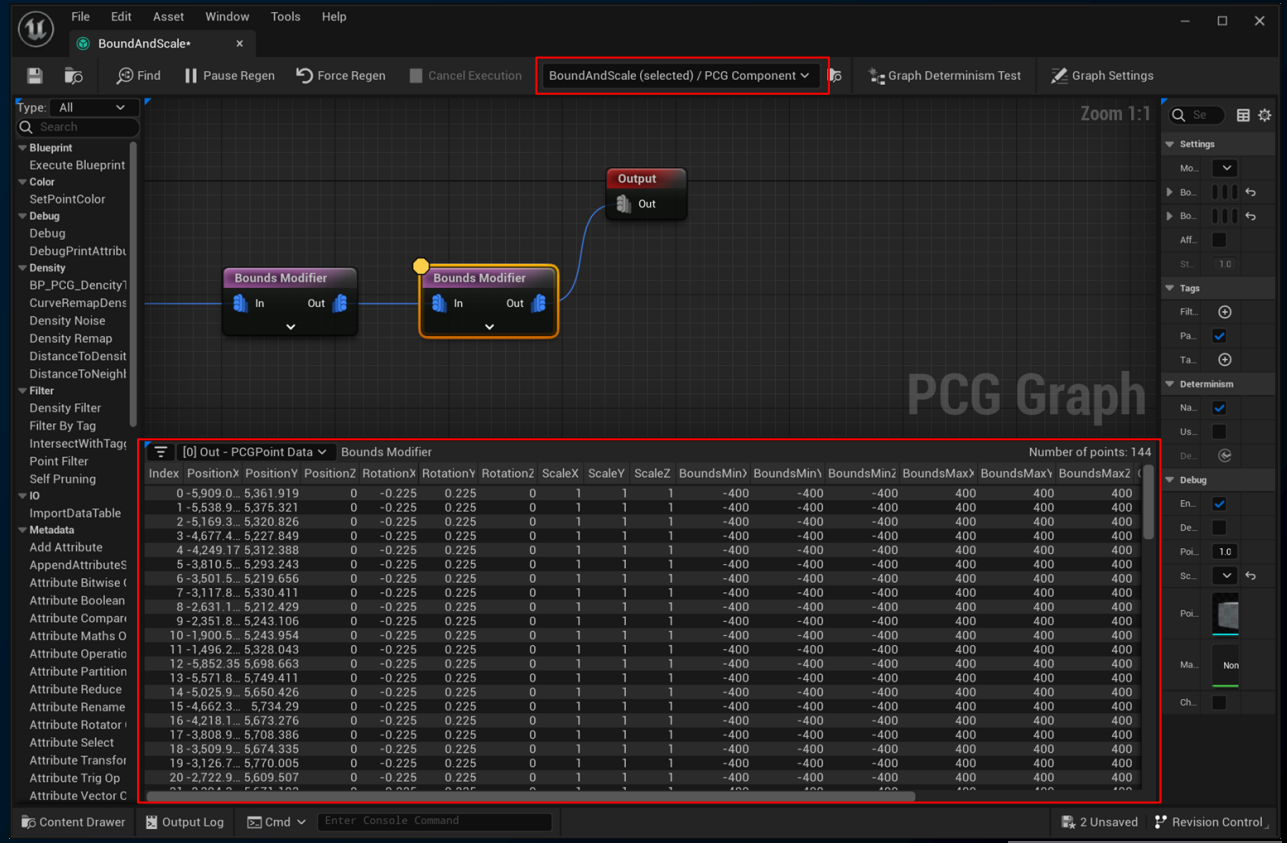
- PointMesh
- Material Override





デバッグ

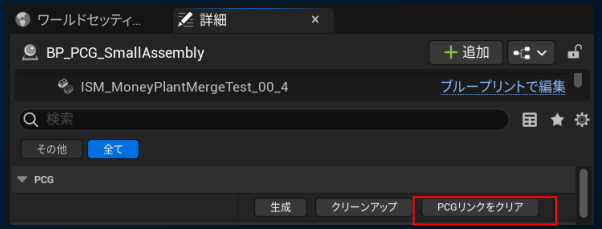


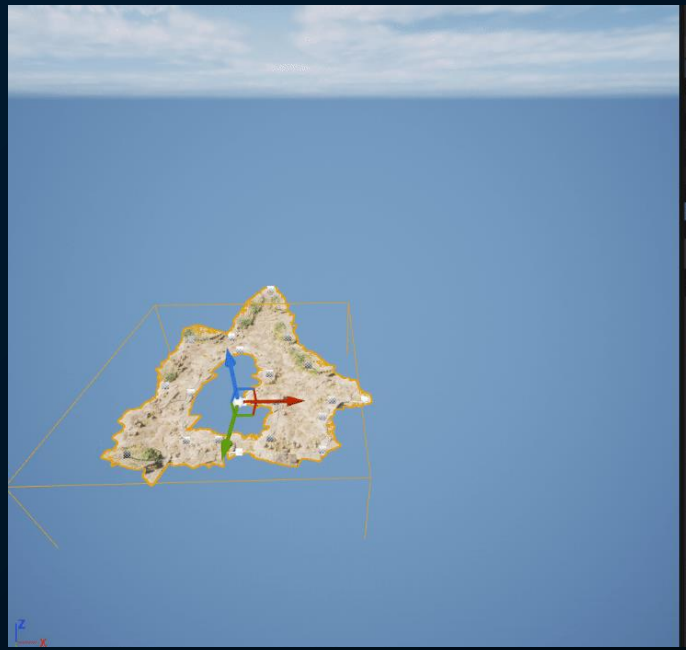


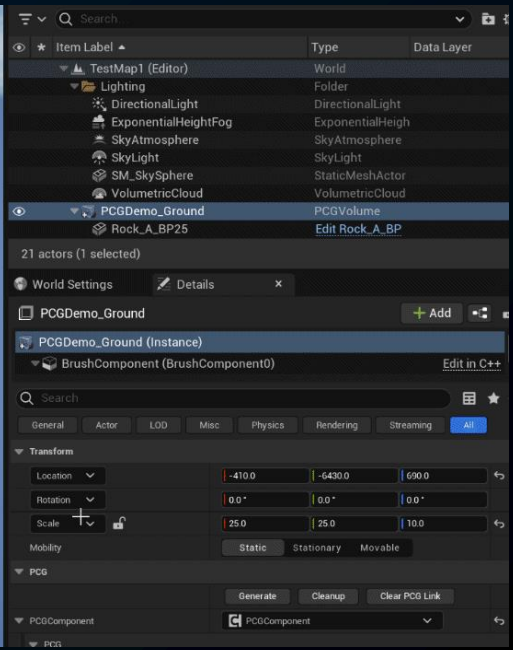
UNREAL ENGINE

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





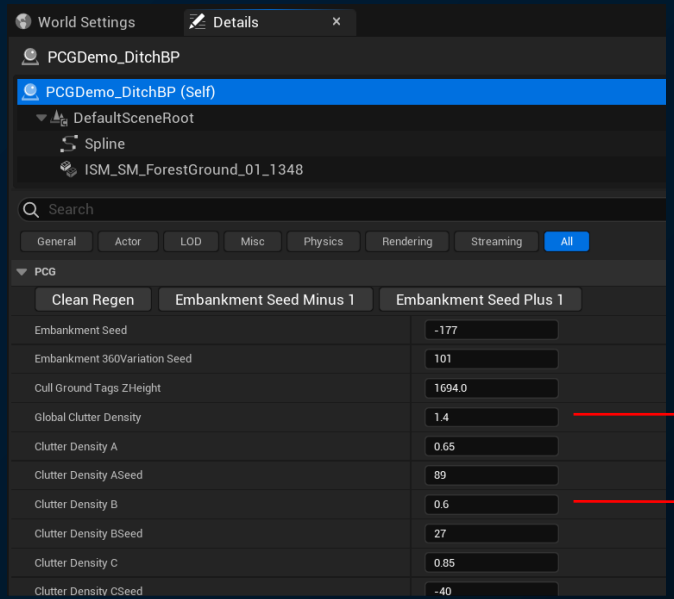


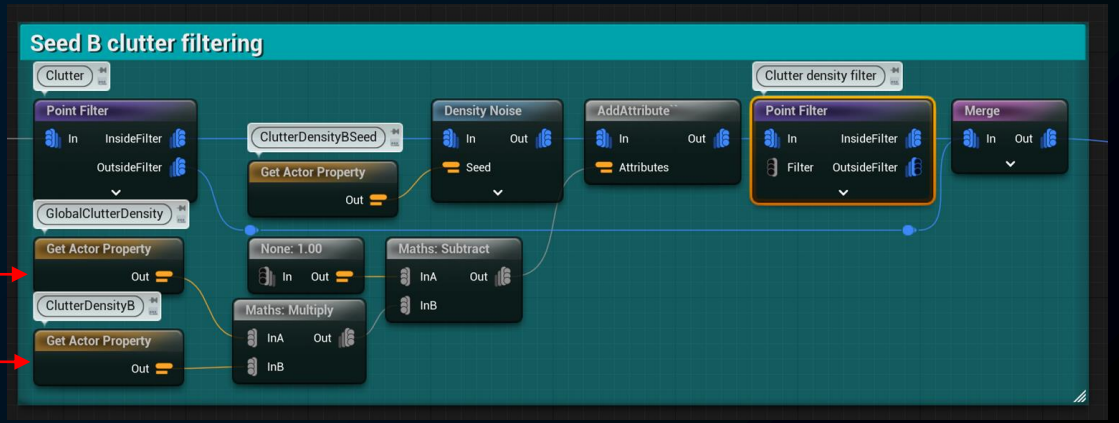
ランタイムでの動作

- 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



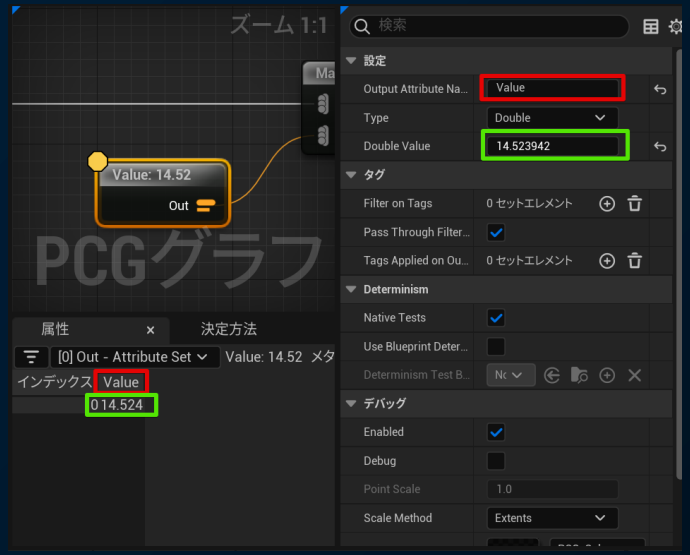


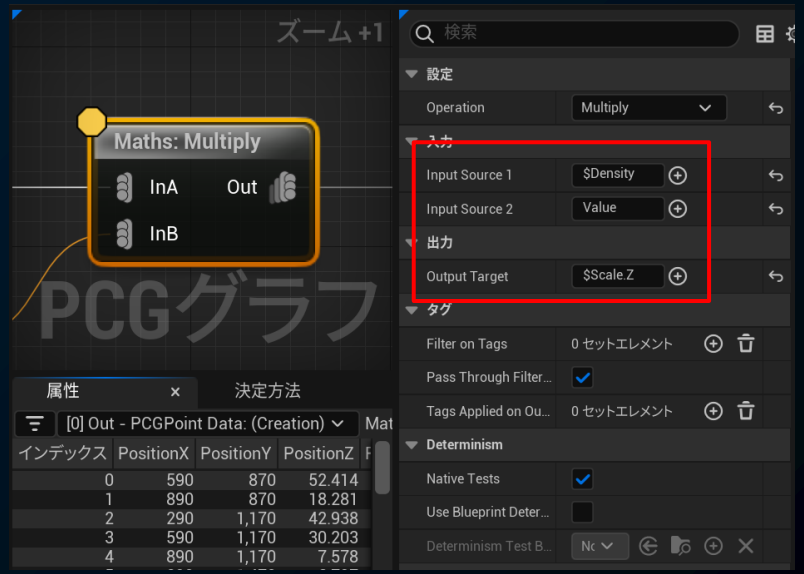
UNREAL ENGINE

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(flaot)
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

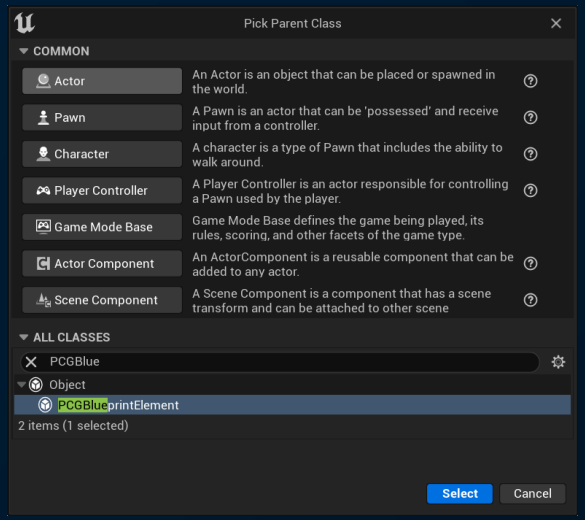
プロパティとアトリビュート

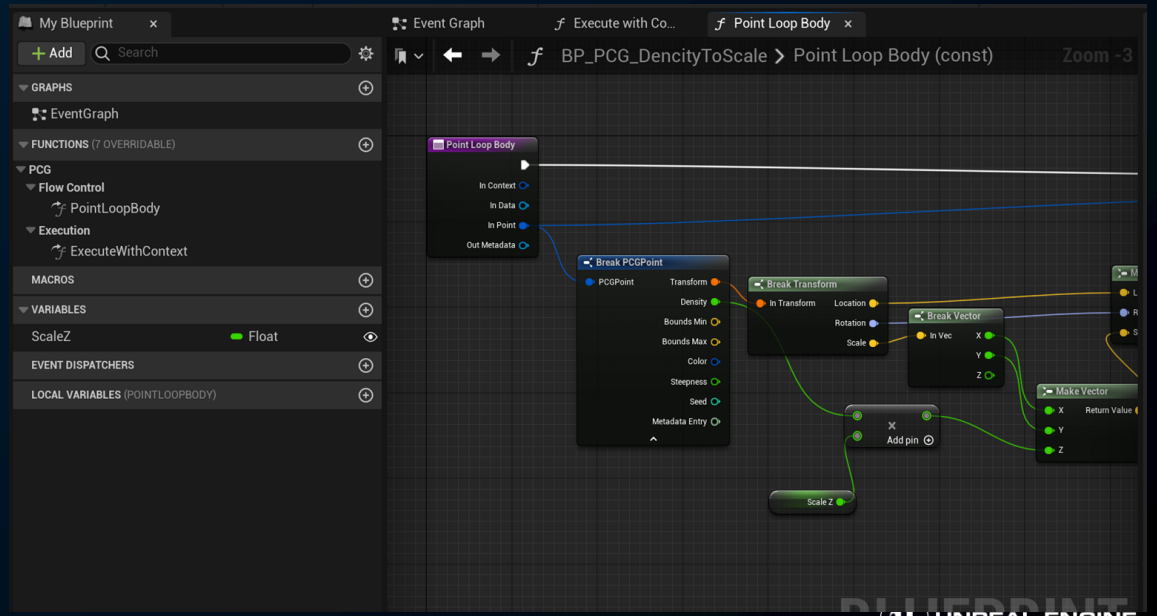




PCG Element

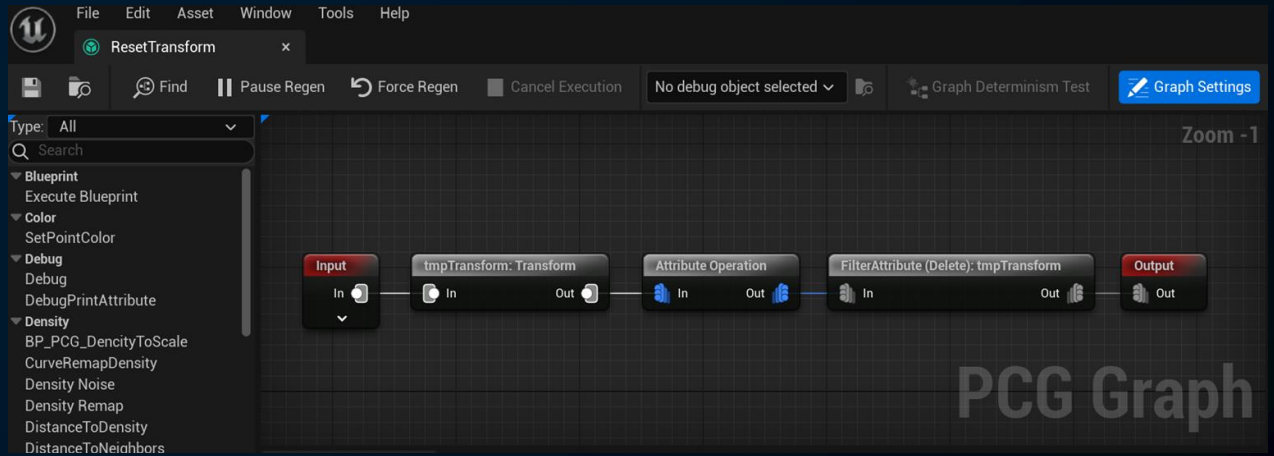
- PointLoopBodyはマルチスレッド動作
- ブループリントノードを使って点群内の各ポイントを加工



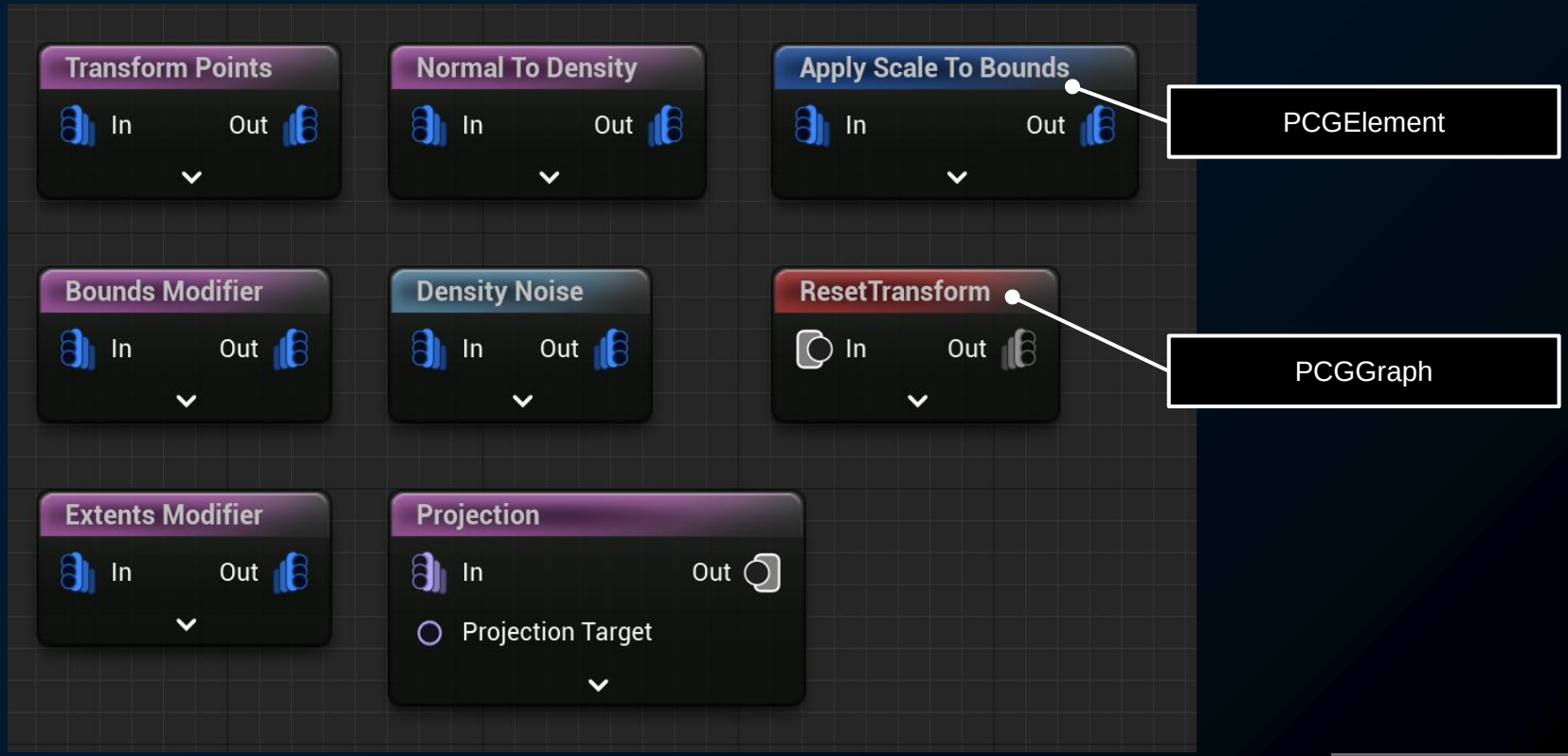


PCGGraphのマクロ的な利用

- 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

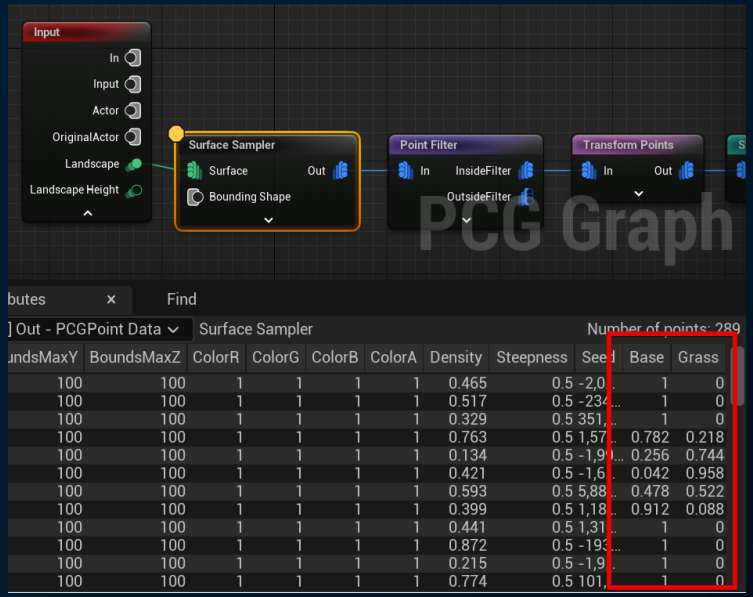


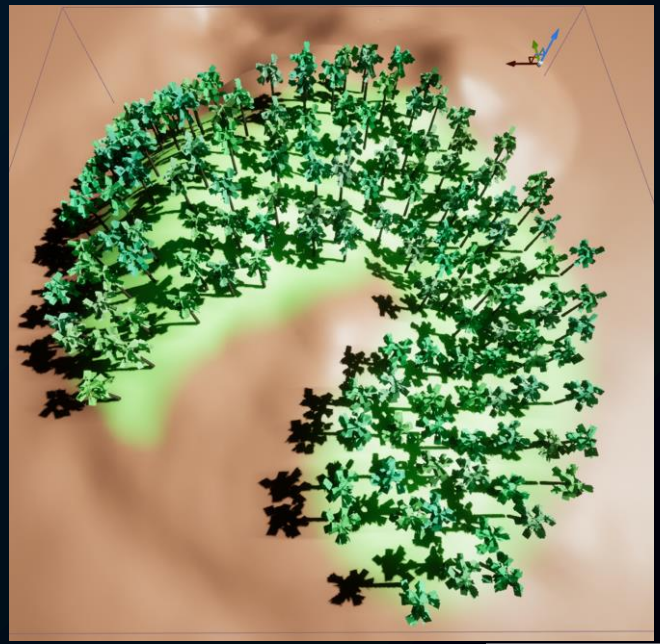
PCGElementとPCG呼び出し



Landscape layer

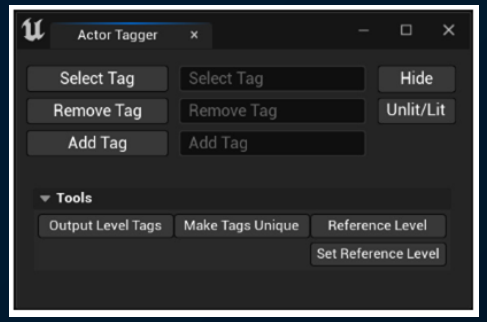
5.2では 1x1 Sections Per Componentのみ対応
2x2 section対応 は5.3から(CL-25796774)

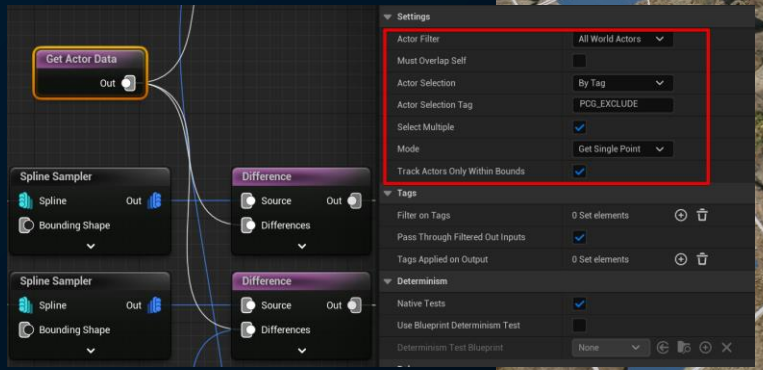


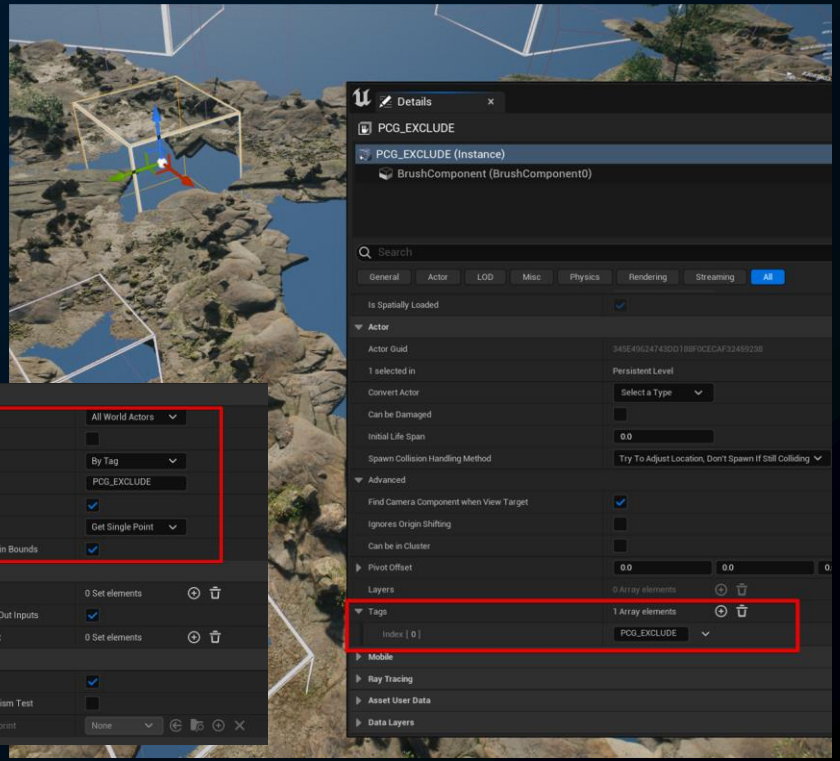


Tag

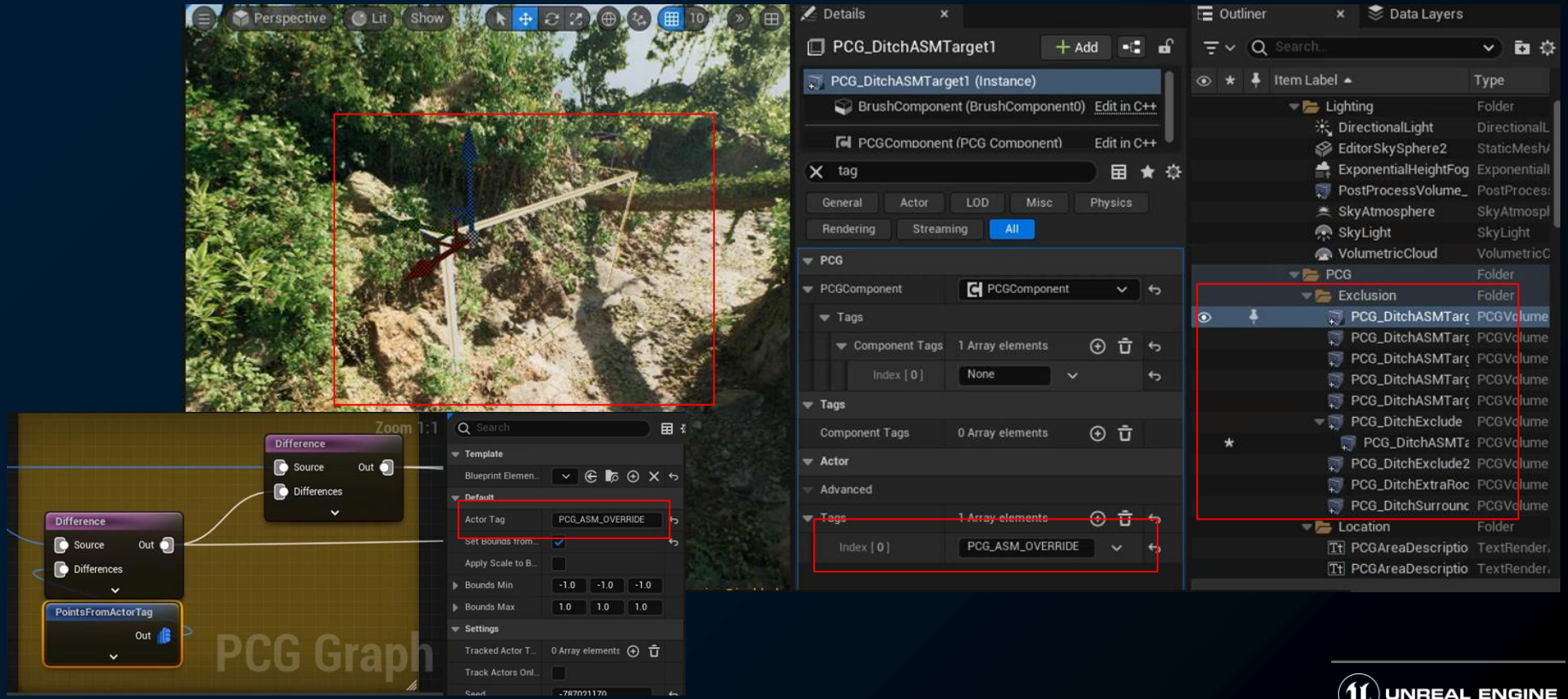
- 除外設定などにActorTagが便利
- ActorTaggerユーティリティ
  - /Game/PCG/Utilities/ActorTagger/EUW_ActorTagger





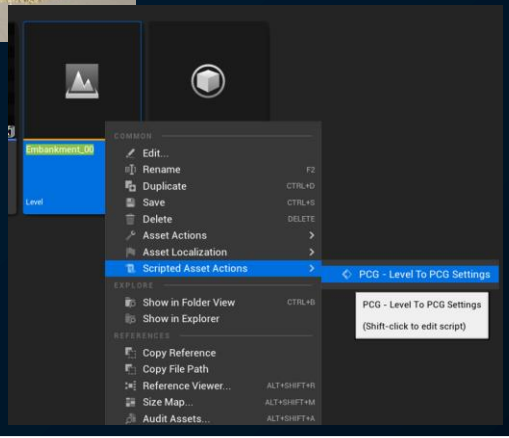


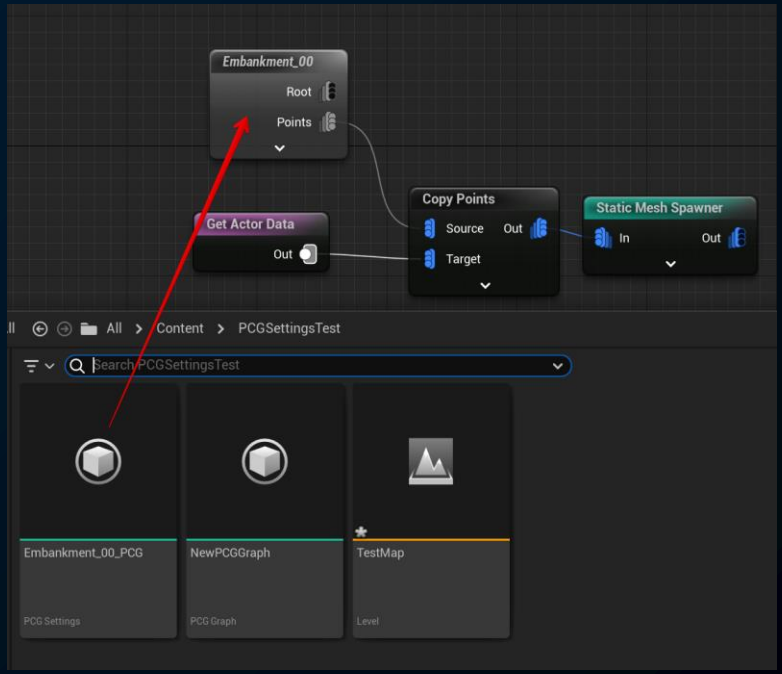
Tag



UNREAL ENGINE

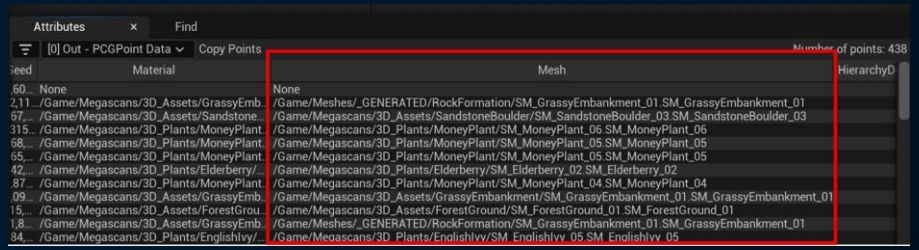
PCG Settings

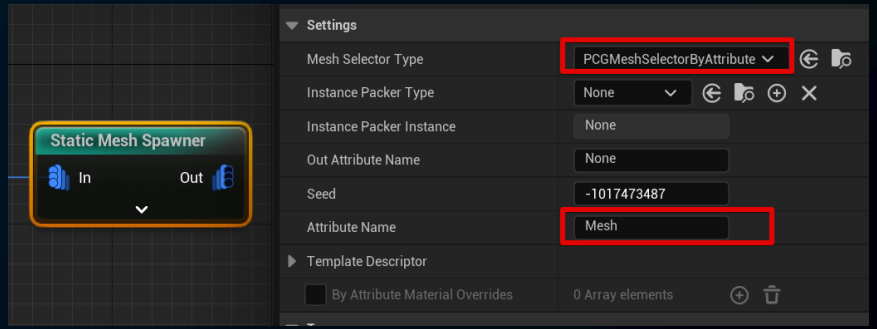




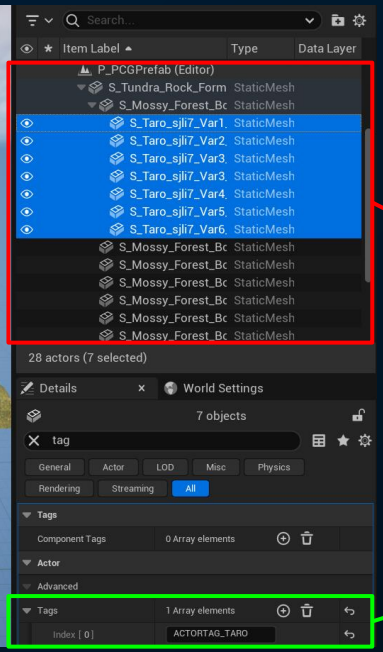
PCG Settings

- 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
- PCGMeshSelecterByAttributeを使ってデータ中のメッシュ名で生成



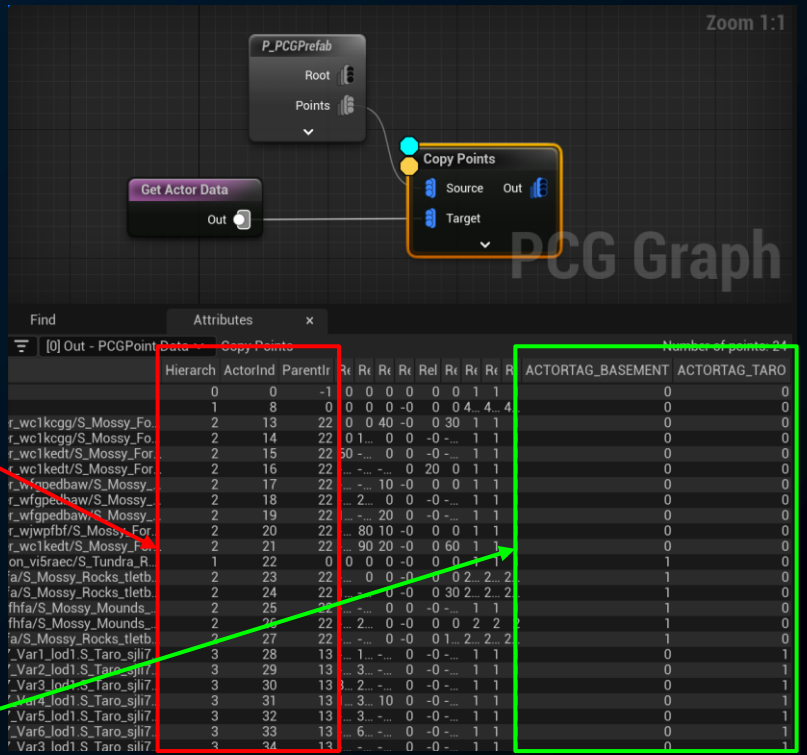


PCG Settings



親子構造

ActorTag



UNREAL ENGINE

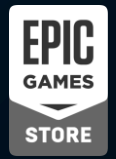
参考リンク

- 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
  - [カスタムライセンシー様向け]
    - 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


EPIC
GAMES