UNREAL
ENGINE

Unreal Fest West 2018

Fortniteを支える技術

Epic Games Japan – Yutaro Sawada

資料は後日公開します

目次

- Fortniteについて
- Fotnite Battle Royaleを実現するために
  - ライティングとシャドウについて
  - LODとHLOD
  - アニメーション
  - Significance Manager
- まとめ
- 付録

Fortniteについて

3月に日本で配信開始。
TPSのアクションビルディングゲーム

Fortniteについて

大きく分けて2つのモードがあります

- Save the world(世界を救え)
  - オンラインで協力して敵と戦う
- Battle Royale(バトルロイヤル)
  - 100人対戦
  - プレイヤー同士で戦う

Battle Royaleの話がメイン

Fortnite Battle Royaleの特徴

- 100人が同時に
  ネットワーク上で対戦
- 2.5 km x 2.5 kmの
  広いマップ
- 建物を壊したり、
  資材を集めて、
  新しく建築することができる

Fortnite Battle Royaleの特徴

マルチプラットフォーム

Windows, Mac, PS4, XBOXONE, iOS, Android...
(Android版は現在開発中で未リリース)

- ハードウェア毎にスペックがバラバラなので
  最適化が必要に
  - 描画のクオリティやエフェクトなど、
    スケーリング可能なパラメーター設計が必要になった

今日話すこと、話さないこと

- 話すこと
  - 描画周りが中心
    - ライティング
    - アセットリダクション
    - アニメーション
- 話さないこと
  - ネットワーク
  - AI

Fotnite Battle Royaleを実現するために

課題

100人対戦かつ、建物の破壊や建築といった
コンセプトを実現させる

見た目を保って如何に処理を端折るか

Fotnite Battle Royaleを実現するために

- ライティング
- アセットリダクション
- アニメーション
- Significance Manager

ライティング

Battle Royaleの特徴①

建物などのフィールド上のモノを壊せる
昼や夜、ストームなどライティングが変わる

UNREAL ENGINE

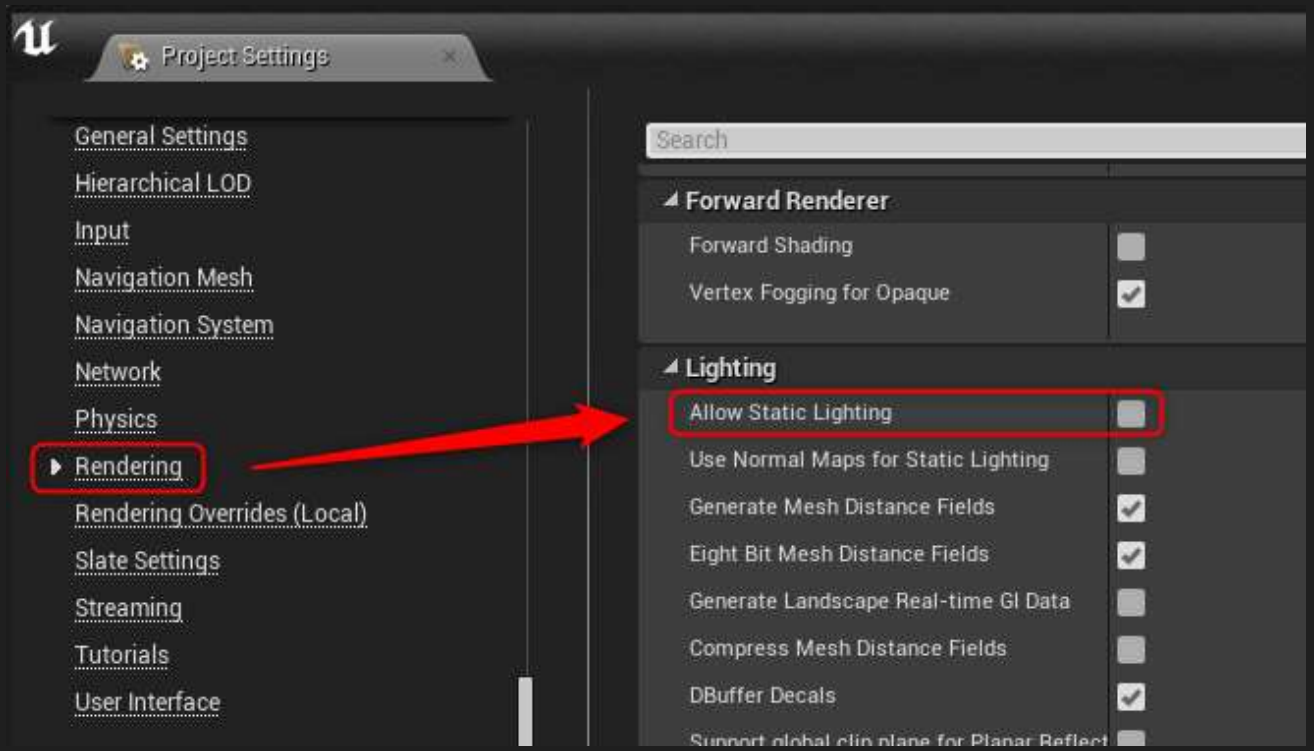

1:14
27
0
35
30
0
0 100
84 100

事前計算した静的なLightingが使えないことから

すべて動的なライティング

Static Light(静的なライト)をオフにする方法

- Project SettingsのRenderingから、
  Allow Static Lightingのチェックをオフ



シャドウについて

Cascade Shadow Map(CSM)

×

Ray-traced distance field shadows(RTDF)

Cascade Shadow Map(CSM)について

UE4で通常使用しているシャドウ

○ 利点

- 背景に配置されている静的オブジェクトから、アニメーションしている動的なオブジェクトまで、静的/動的関係なくオブジェクトのシャドウを動的に生成することが可能
- 品質がそこそこ良い

△ 注意点

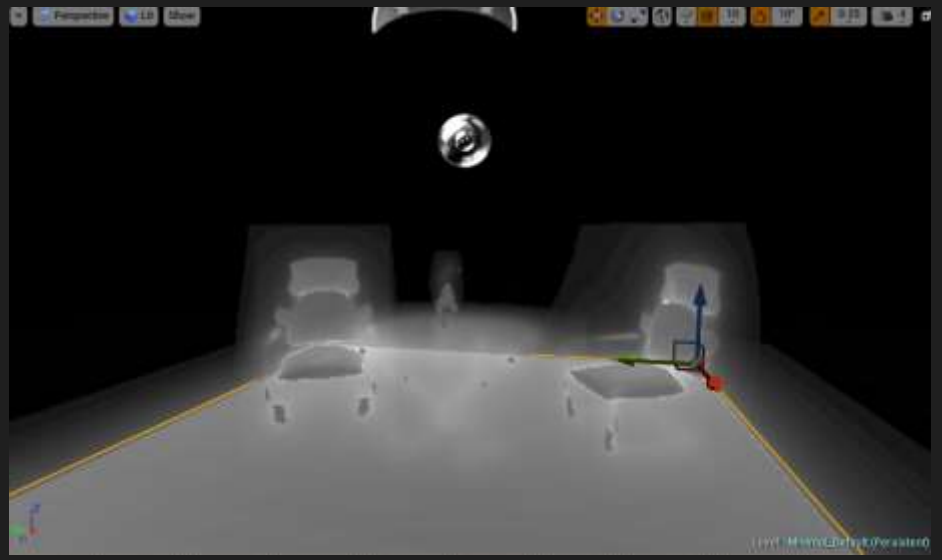
- 描画にコストがかかる

Ray-traced distance field shadows(RTDF)について

事前計算した3Dテクスチャを使用するシャドウ

利点

- CSMよりもコストが安く描画できる
  - CSMより30%～50%



Ray-traced distance field shadows(RTDF)について

注意点

- エディタ上で事前計算しメッシュがデータを保持する(Ambient Occulusionでも利用する)
- 歪んだスケールやアニメーションは使用できない
- 細いオブジェクトだとレイが抜けて影がきれいにでない
- メッシュ毎にメモリのコストが必要

シャドウについて

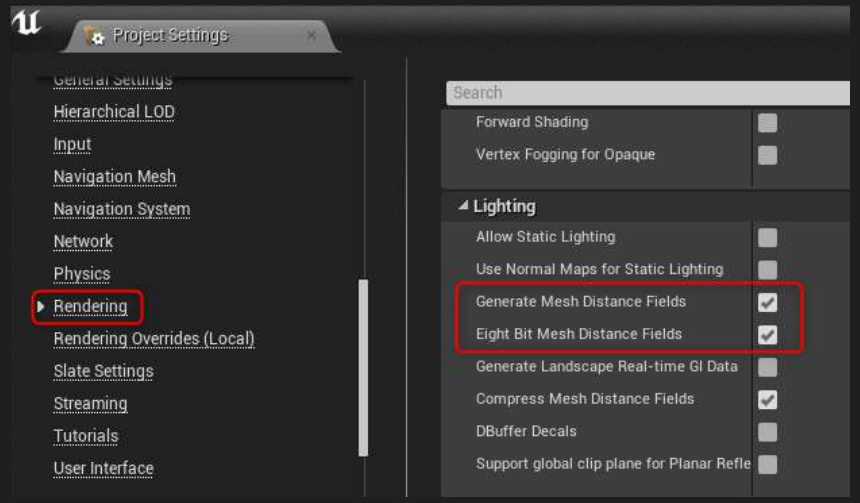
両方の利点を活かし、

- 手前のモノはCSM
- 遠くのモノはRTDF

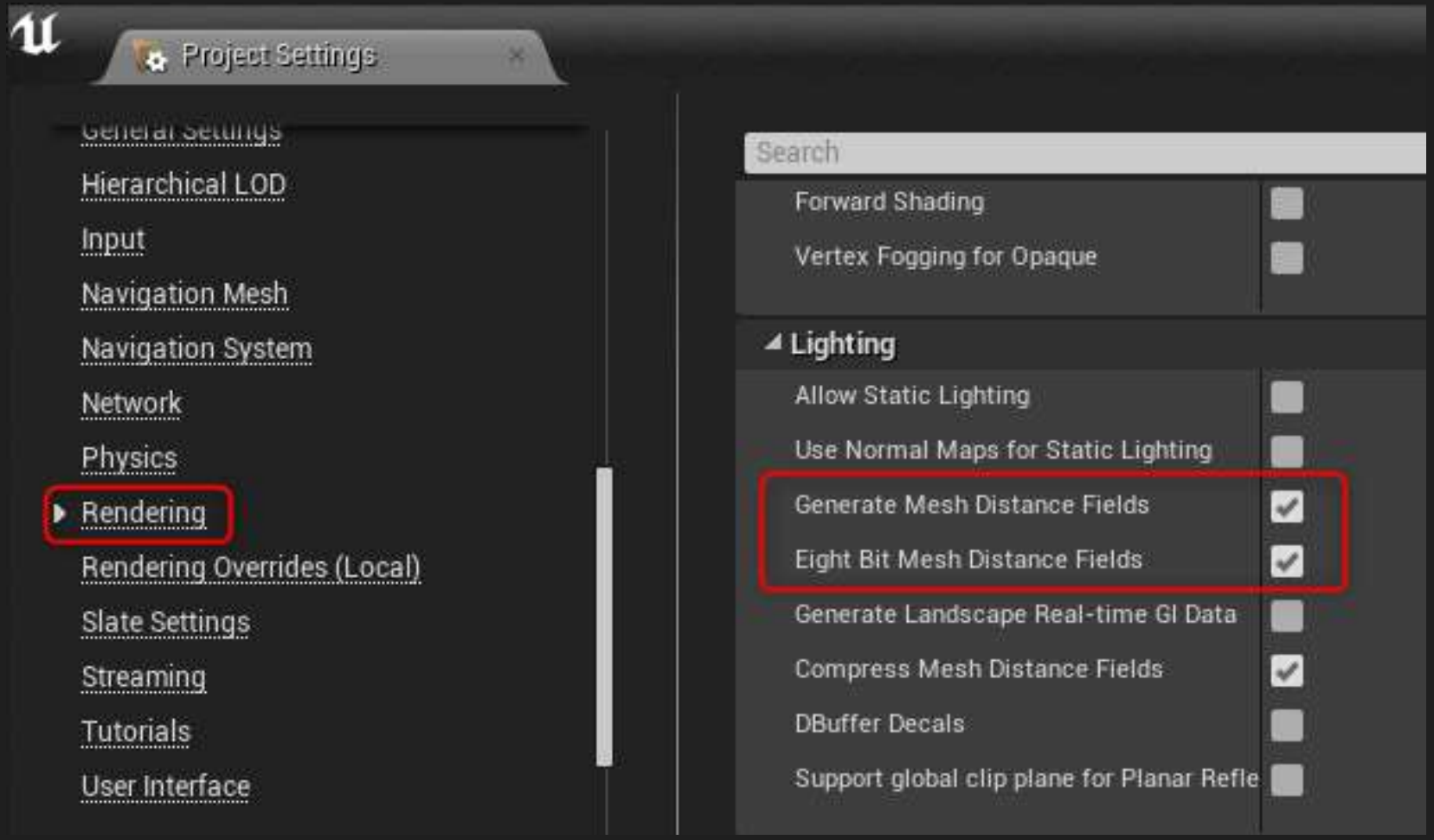
RTDFの使い方

RTDFのセットアップ

- まずProject SettingsのRenderingから、Generate Mesh Distance Fieldsを有効にする
- Eight Bit Mesh Distance Fieldsを有効にすると保存データが16bitから8bitになる
  - シーンの規模などによっては アーティファクトが発生する可能性があるが、データサイズは単純に半分になる

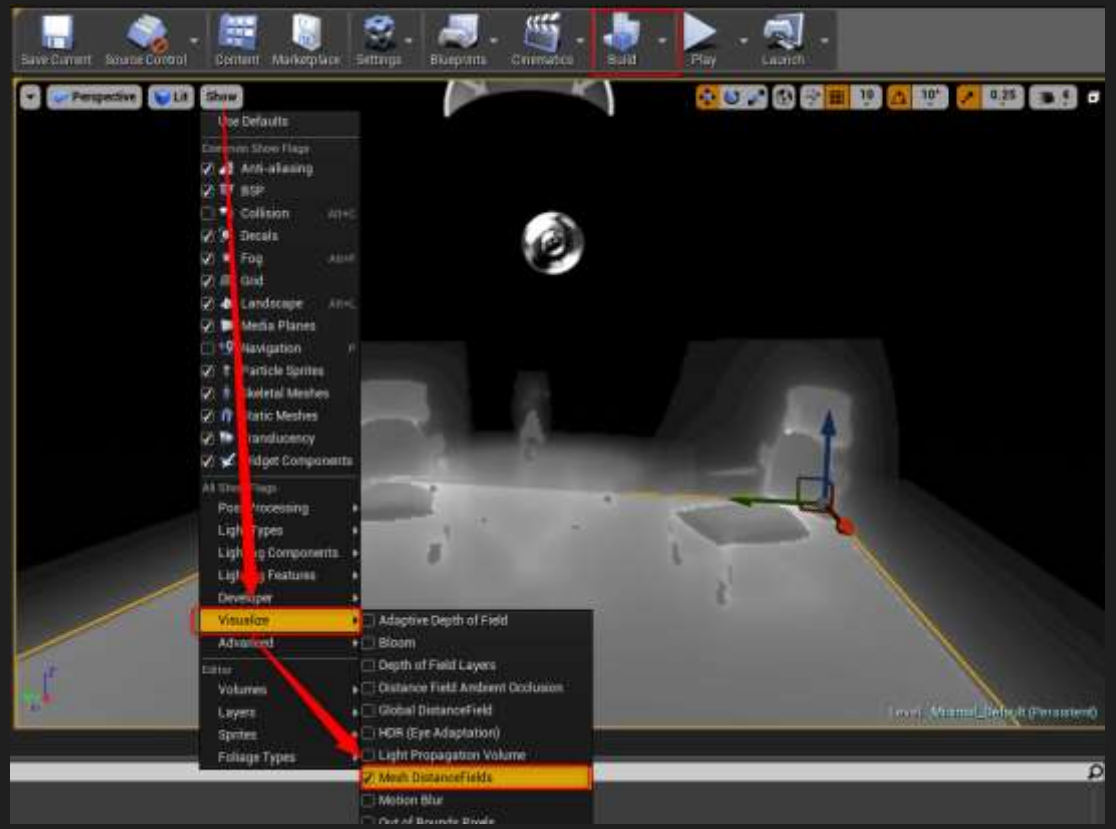


RTDFのセットアップ



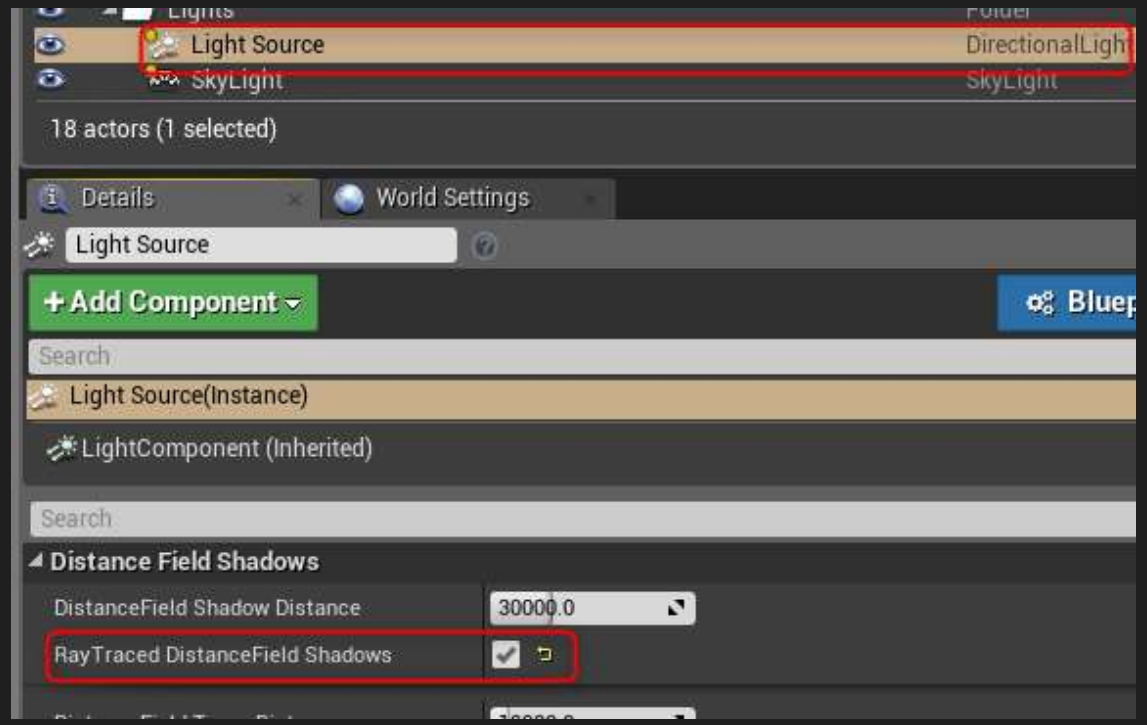
Mesh Distance FieldsのVisualize

- 設定後、Editorを再起動し、
  レベルをビルドすると、
  事前計算データが作られます
- ShowのVisualizeから確認できます



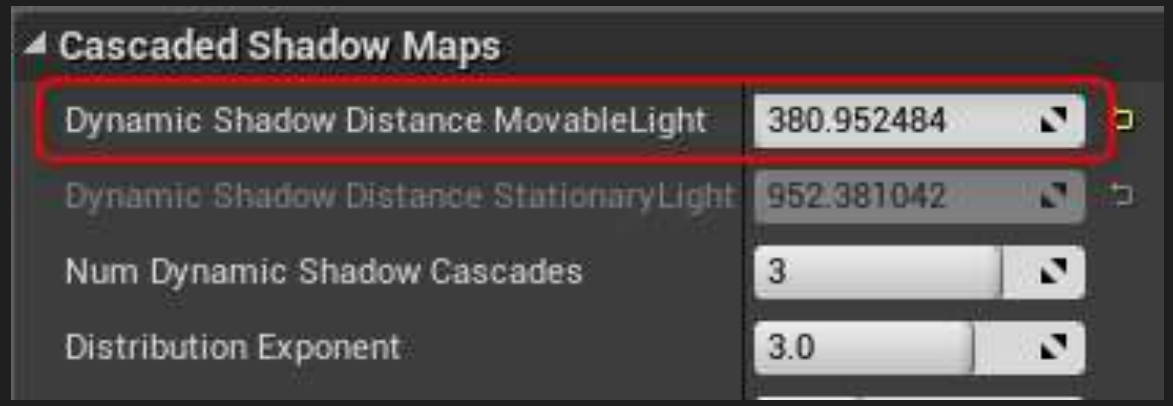
RTDFの使用方法

Directional LightのDetailsから、Distance Field Shadows欄の
RayTraced DistanceField Shadowsのチェックをいれると有効



RTDFの使用方法

Directional LightのDetailsから、Cascaded Shadow Mapsの
Dynamic Shadow Distance MovableLightで影響範囲を指定



ライティング終わり

アセットリダクション

Battle Royaleの特徴②

フィールドが広く建物や木といった
オブジェクト数が多い

上空から飛び降りるアクションがあり、
フィールドを見渡すことができる

UNREAL ENGINE

GPU負荷や、DrawCallを減らす改善が必要

LODとHLOD+Impostorによって改善

Level of Detail(LOD)

- 遠い場合ポリゴンの
  少ないモデルに切り替えて
  描画負荷を減らす手法

LODの注意点

利点

- 描画時のポリゴン数が減り、GPU負荷が軽減
  - (Early Z-pass, Shadow pass, Base passに効果的)

注意点

- LODを保持すれば保持するほどメモリ量は増える

LODを使用するには

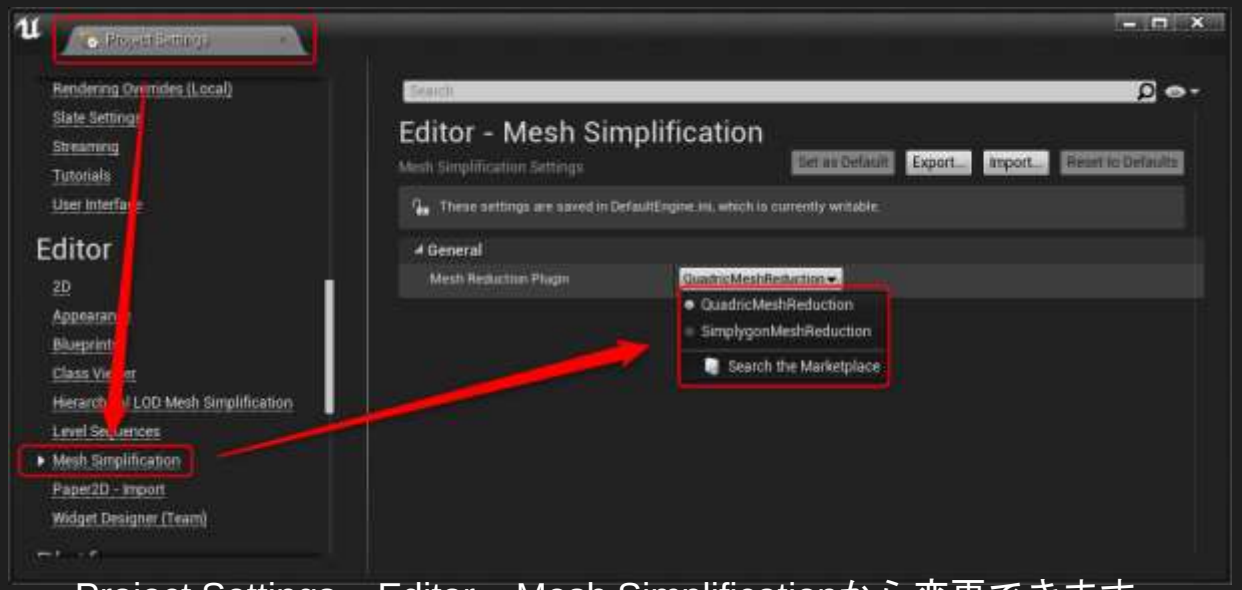
- UE4でLODを使用するには2つの方法があります。

  1. 外部ツールで作成し、インポートし使用する方法

  2. UE4内部でLOD用メッシュを作成し使用する方法

UE4でLOD用メッシュの作成

- UE4では、2つのLODメッシュ作成方法があります。
  - UE4内製ツールで行う方法(QuadricMeshReduction)
  - ミドルウェアのSimplygonで行う方法(SimplygonMeshReduction)

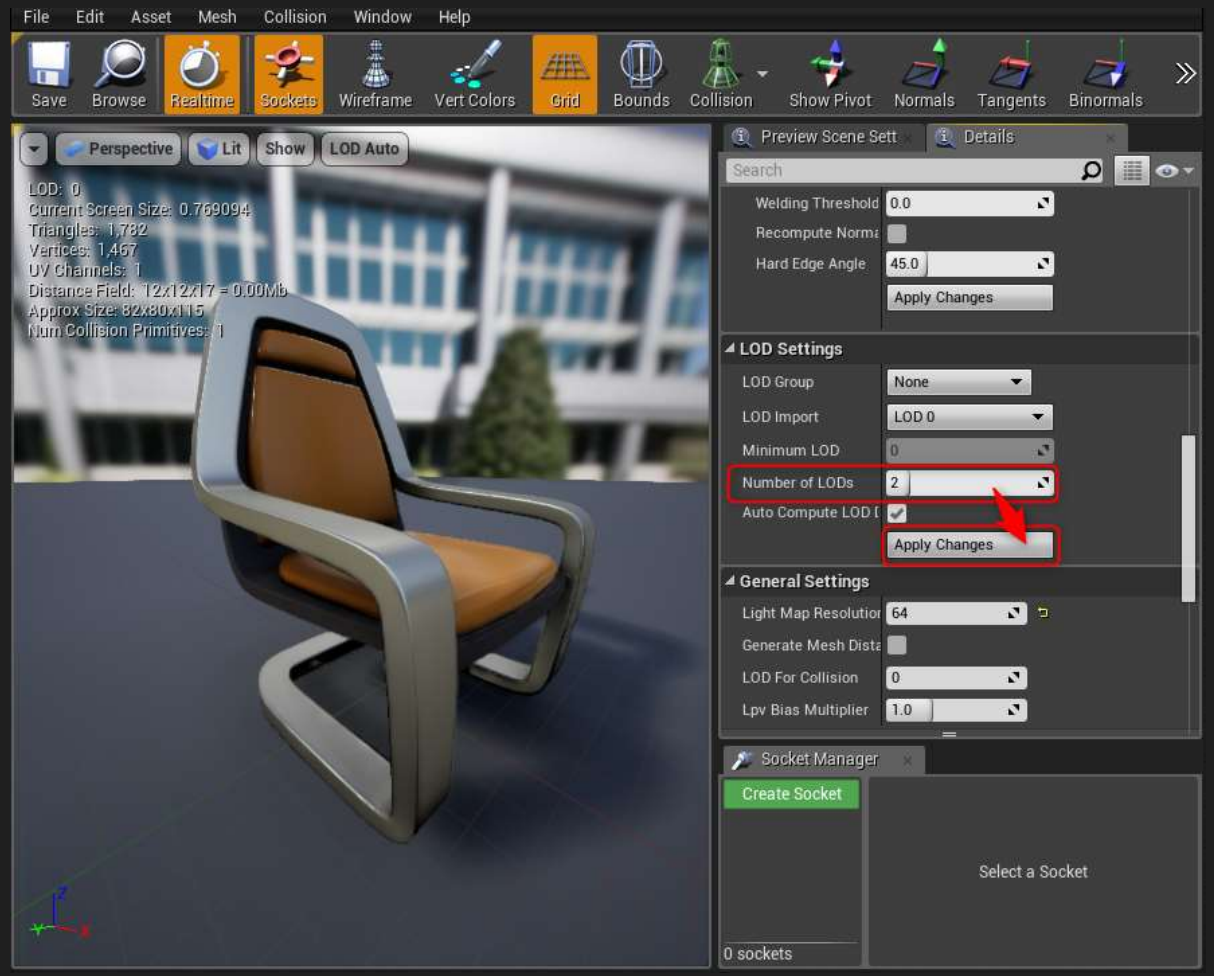


Project Settings – Editor – Mesh Simplificationから変更できます。

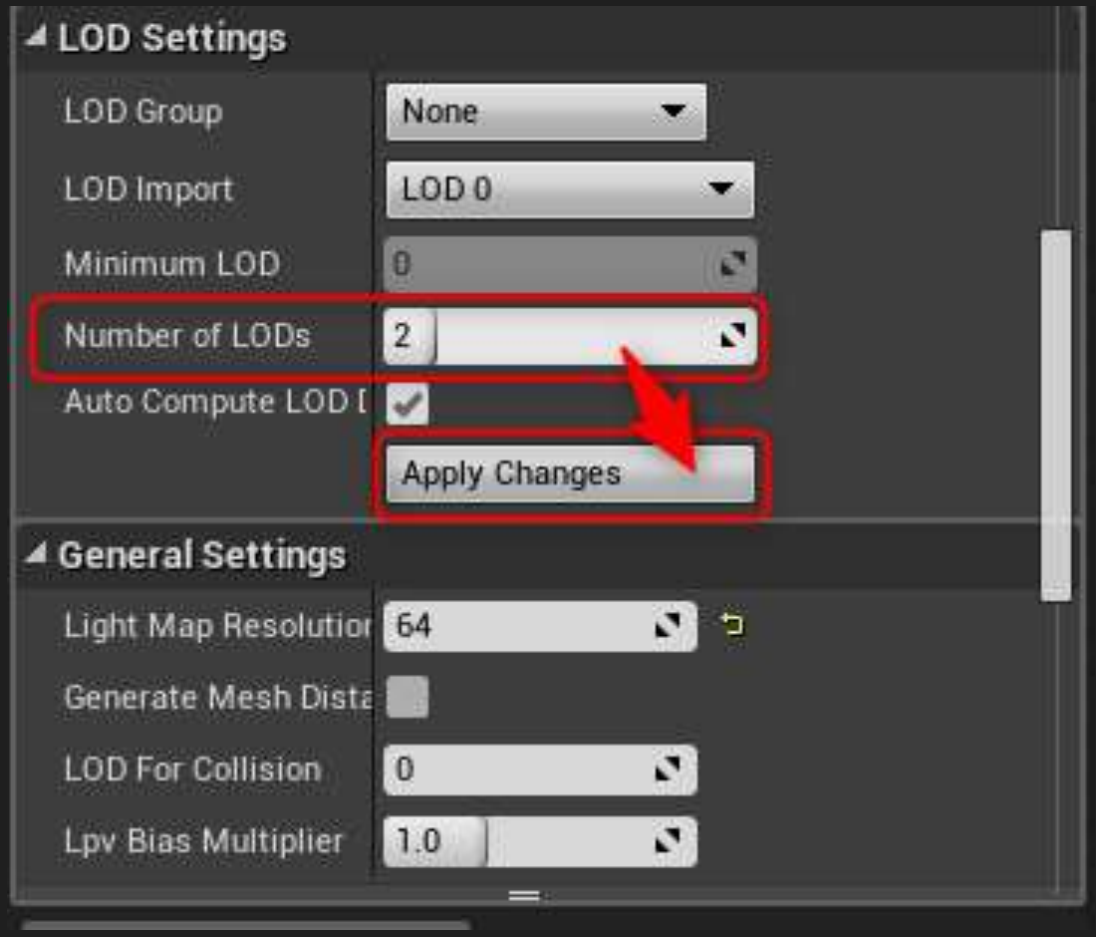
LODの作成方法

LODの追加方法

Meshを開いて、LOD Settingsから
Number of LODsの数を増やし、
Apply Changesで適応する。

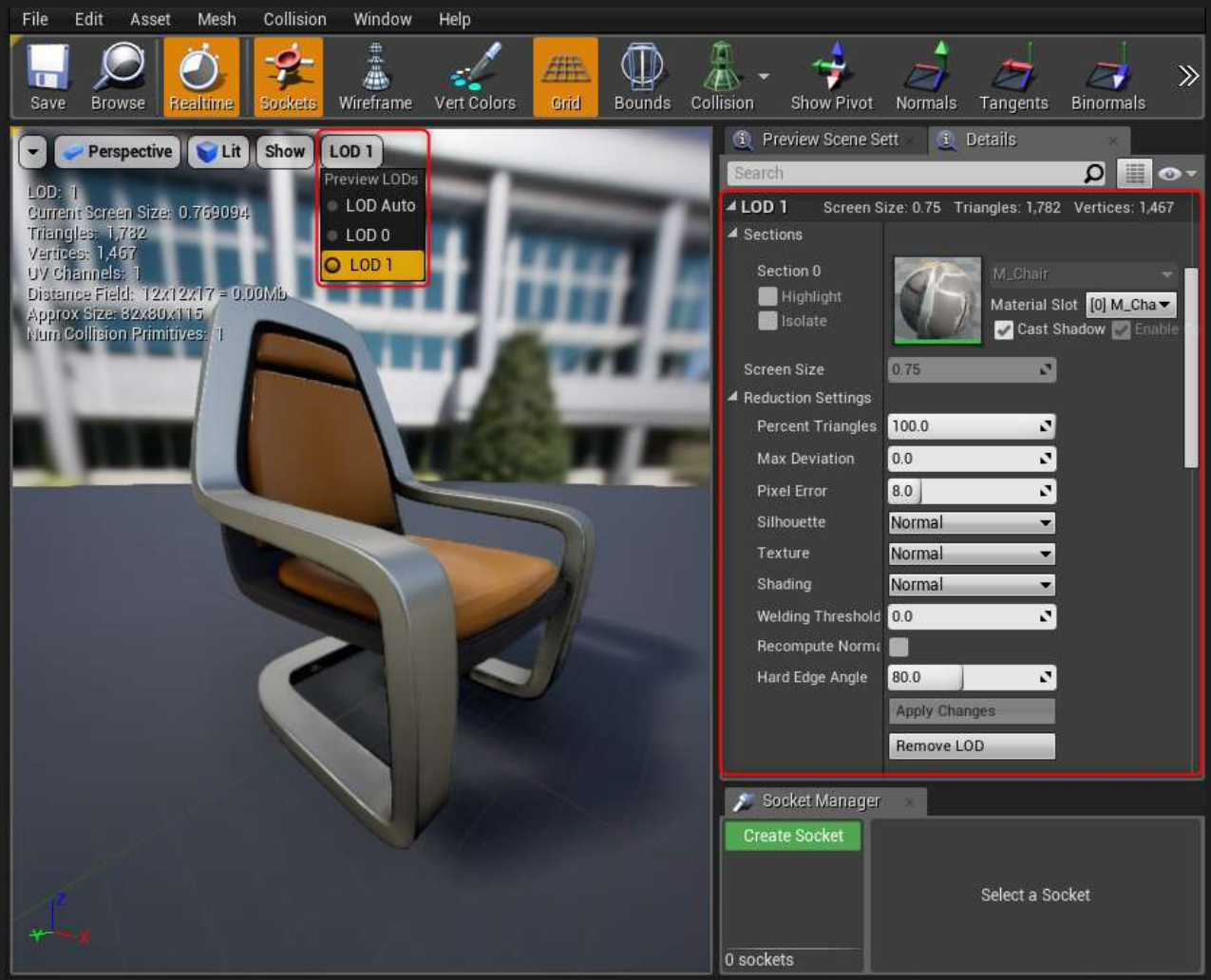


LODの追加方法


LOD Settings
LOD Group
None
LOD Import
LOD 0
Minimum LOD
0
Number of LODs
2
Auto Compute LOD
Apply Changes
General Settings
Light Map Resolution
64
Generate Mesh Dista
LOD For Collision
0
Lpv Bias Multiplier
1.0

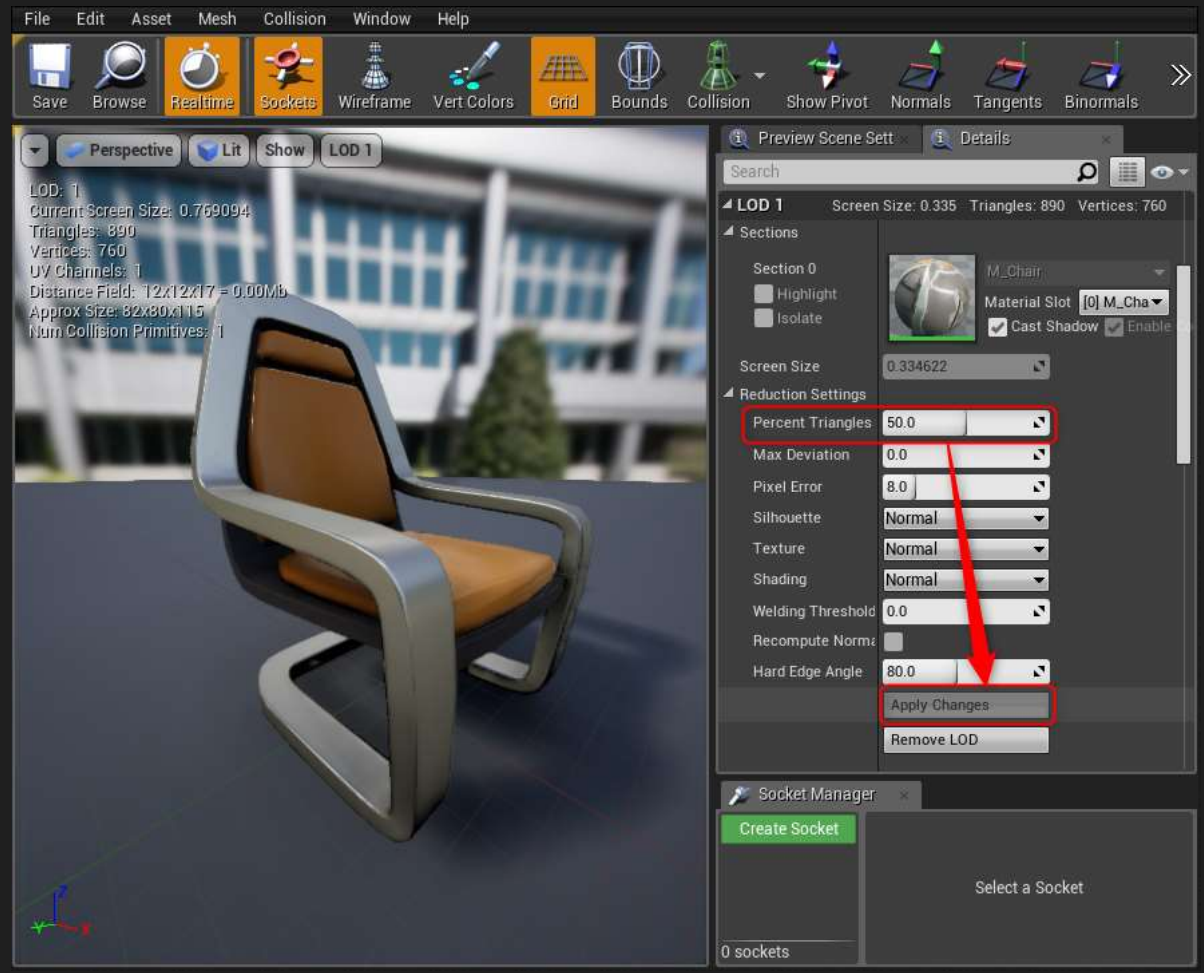
LODの追加方法

LODの表示切り替えが
可能になっており、Detailsから
各種設定が行えるようになる。

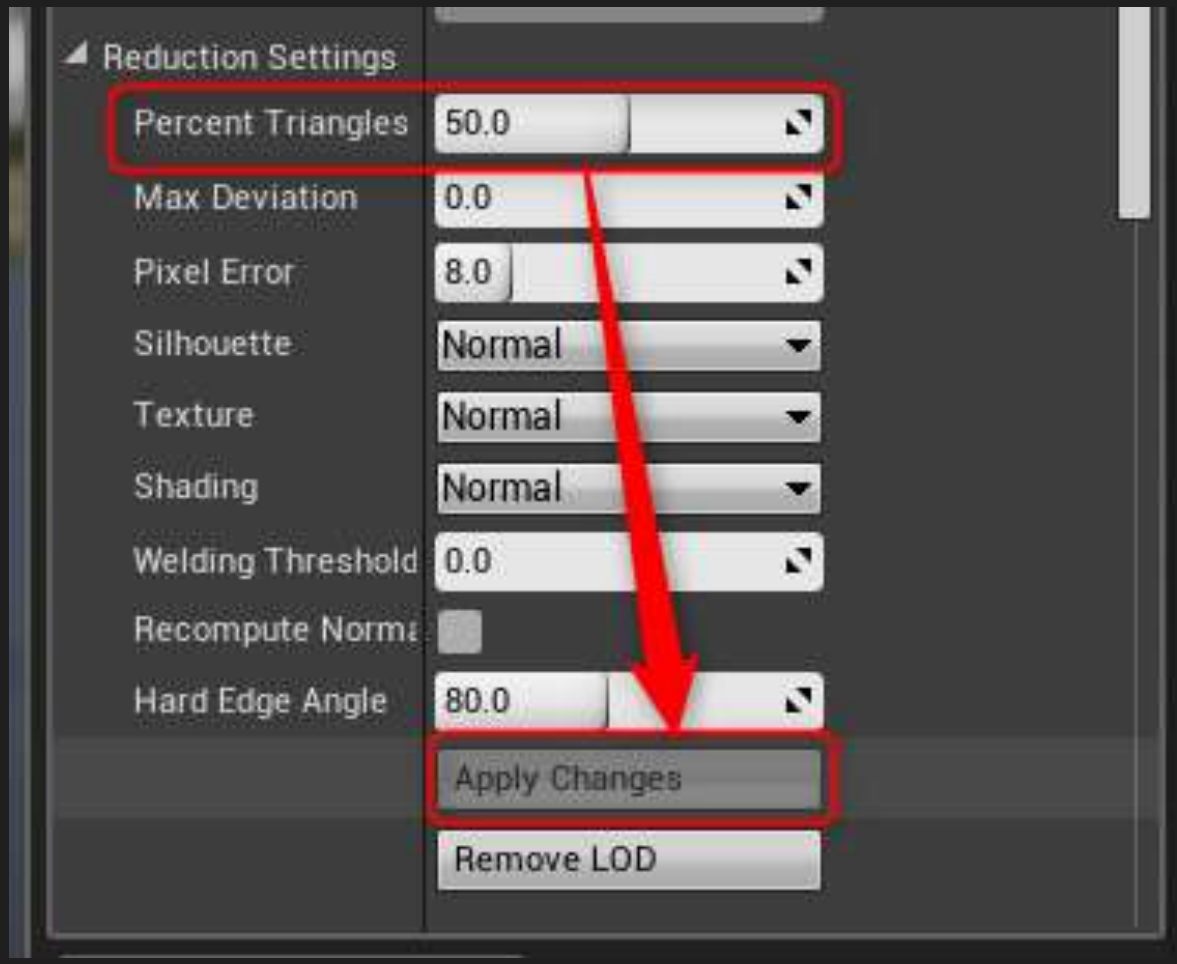


LODのReduction設定

Percent Trianglesを50%に設定、
Apply Changesで適応させる
ポリゴンが半分のLODモデルを
作成することができる



LODの追加方法

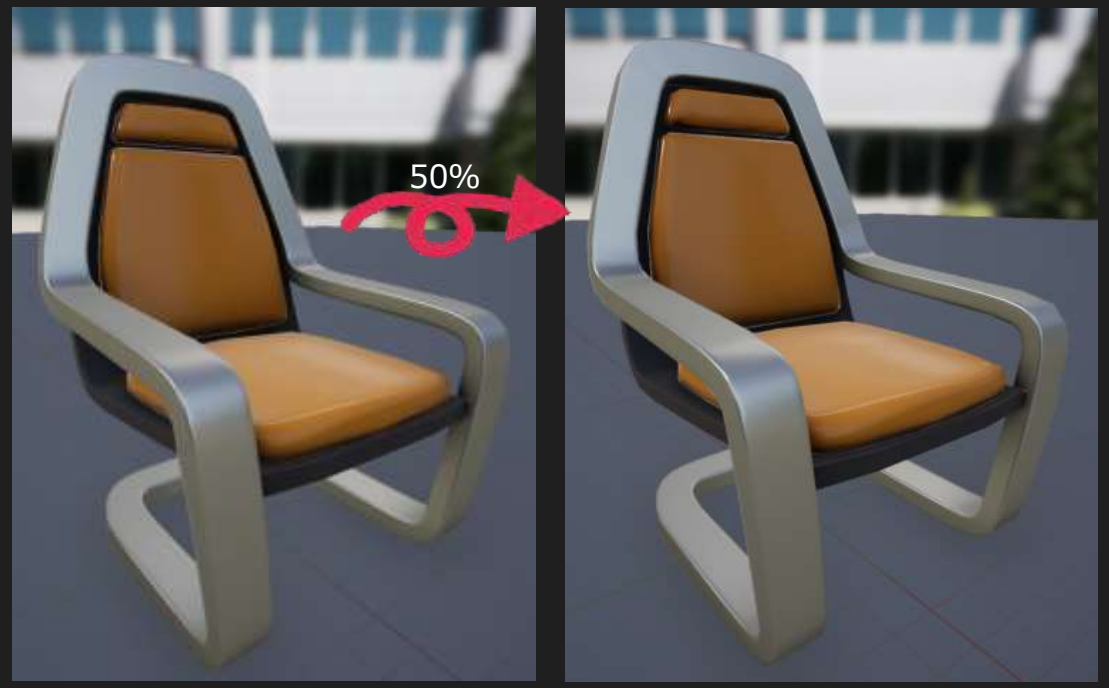


890 triangles, 760 vertices.

Static Mesh-Polygon percentageのみ変更した場合の

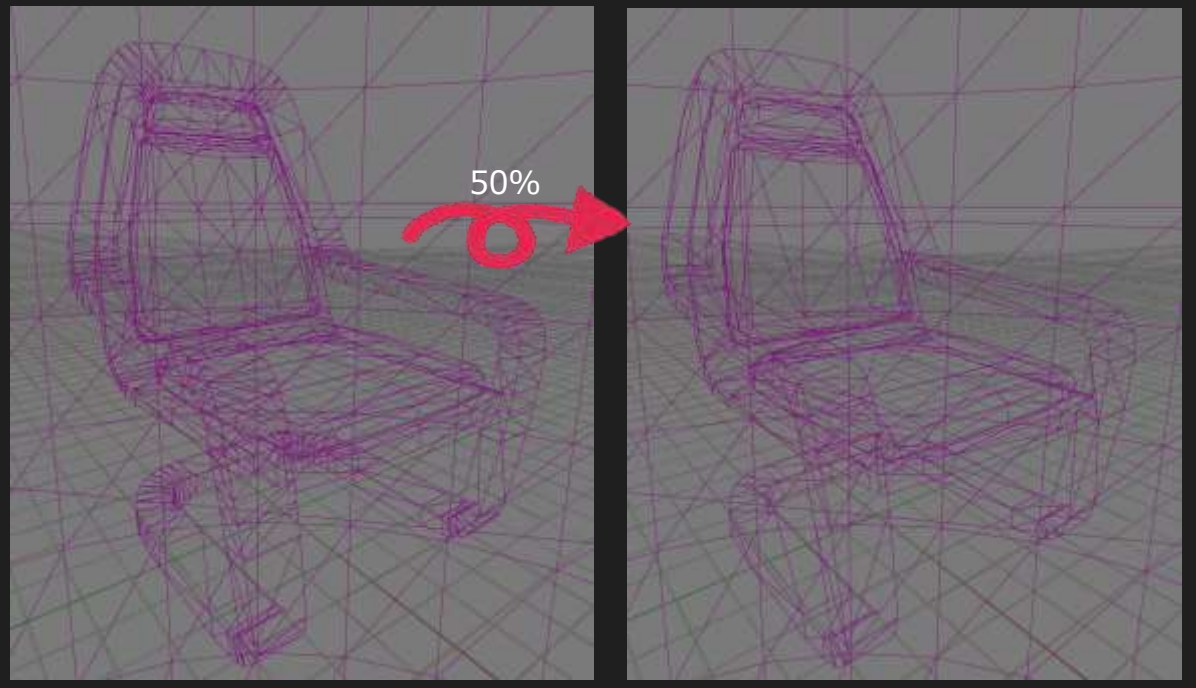
LODの各種比較

Polygon Reduction 50%(UE4内製)



LOD0
1782 triangles, 1467 vertices.

LOD1
890 triangles, 760 vertices.



LOD0
1782 triangles, 1467 vertices.

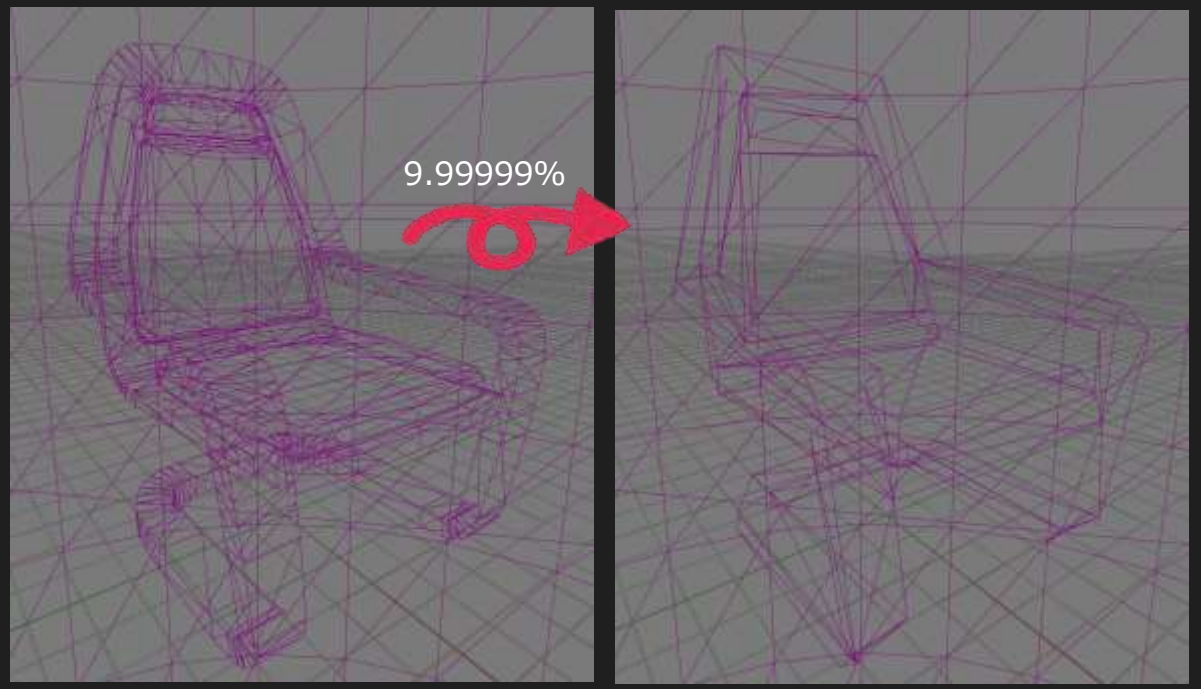
LOD1
890 triangles, 760 vertices.

Polygon Reduction 10%(UE4内製)

※10を入れると9.99999となってしまう

LOD0
1782 triangles, 1467 vertices.

LOD1
178 triangles, 199 vertices.



LOD0
1782 triangles, 1467 vertices.

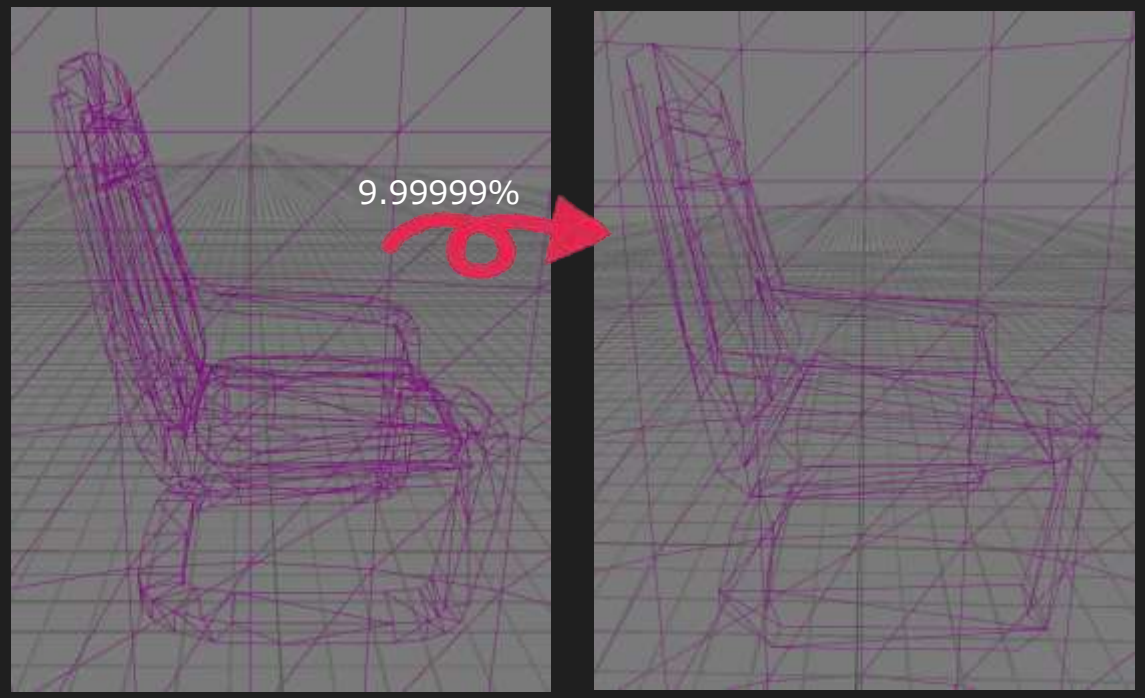
LOD1
178 triangles, 199 vertices.

Polygon Reduction 10%(横)(UE4内製)

※10を入れると9.99999となってしまう

LOD0
1782 triangles, 1467 vertices.

LOD1
178 triangles, 199 vertices.



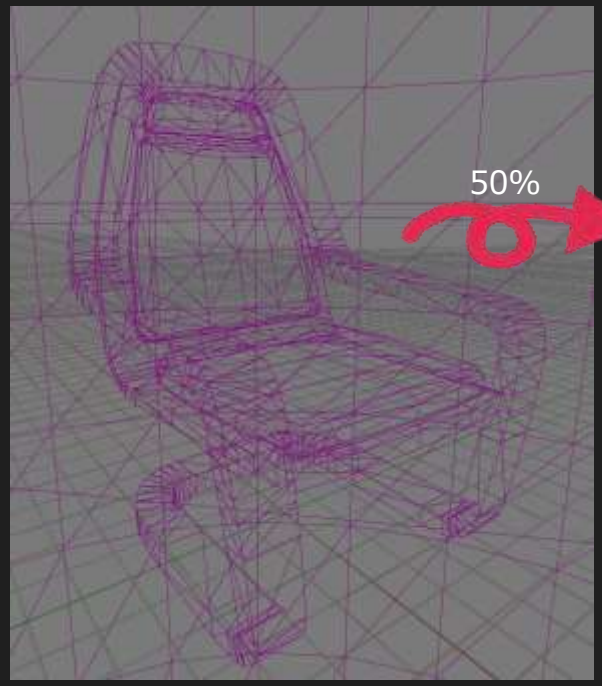
LOD0
1782 triangles, 1467 vertices.

LOD1
178 triangles, 199 vertices.

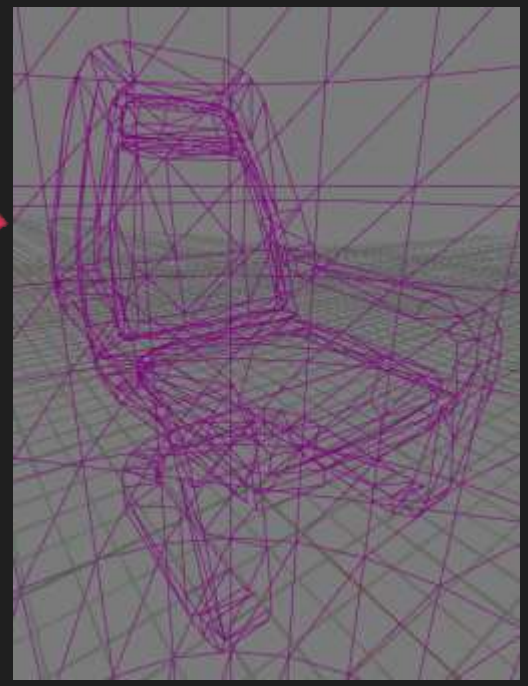
Polygon Reduction 50%(Simplygon)

LOD0
1782 triangles, 1467 vertices.

LOD1
890 triangles, 780 vertices.



LOD0
1782 triangles, 1467 vertices.



LOD1
890 triangles, 780 vertices.

Polygon Reduction 10%(Simplygon)

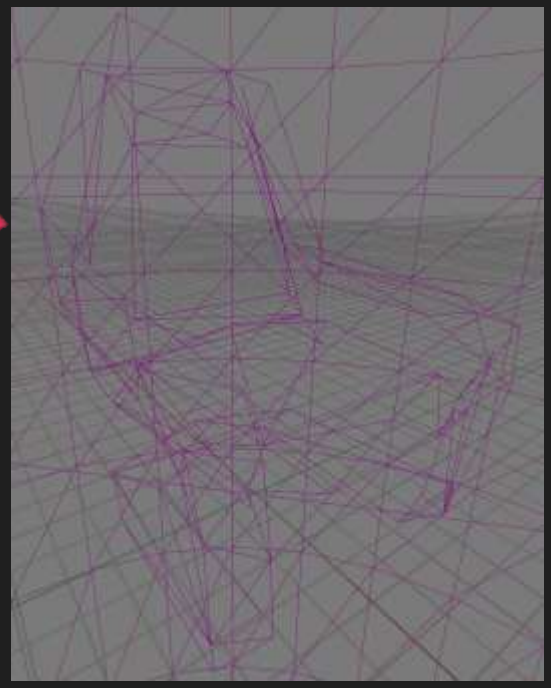
※10を入れると9.99999となってしまう

9.99999%

LOD0
1782 triangles, 1467 vertices.

LOD1
178 triangles, 248 vertices.

9.99999%



LOD0
1782 triangles, 1467 vertices.

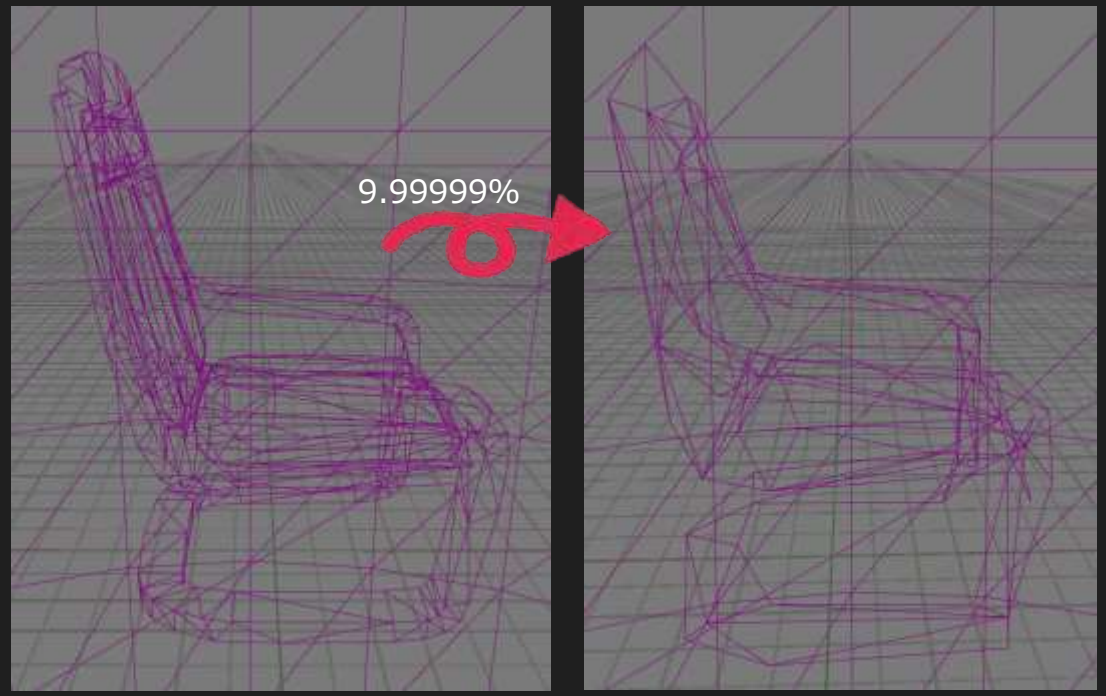
LOD1
178 triangles, 248 vertices.

Polygon Reduction 10%(Simplygon)

※10を入れると9.99999となってしまう

LOD0
1782 triangles, 1467 vertices.

LOD1
178 triangles, 248 vertices.



LOD0
1782 triangles, 1467 vertices.

LOD1
178 triangles, 248 vertices.

LOD比較まとめ

UE4内製

利点

- 大幅に削減してもシルエットが保ててる。
- ポリゴンの数は同じだが、
  頂点数がUE4内製のほうが少ない
- 比較的速い

注意点

- Skeletal Meshのリダクションはできない
- 変更できる設定項目は少ない

UE4内製
178 triangles, 199 vertices.

Symplygon

利点

- UE4のものよりも設定項目が多い
- 細かい設定が可能
  - 調整をすれば良くなる可能性
- Skeletal Meshもリダクション可能

注意点

- ミドルウェアとしてSymplygonを導入する事が必要

Simplygon
178 triangles, 248 vertices.

LOD比較まとめ

比較した結果

- Static MeshのみであればUE4内製のツールで十分良さそう
- サクッとLODモデルを作成する時に有用

元メッシュ
1782 triangles, 1467 vertices.

UE4内製
178 triangles, 199 vertices.

Simplygon
178 triangles, 248 vertices.

Hierarchical Level of Detail(HLOD)

Hierarchical Level of Detail(HLOD)

- 複数の建物、木といった違うアセットのメッシュやマテリアルをマージし、LOD処理を行うことで、描画負荷を減らす方法

HLODについて

利点

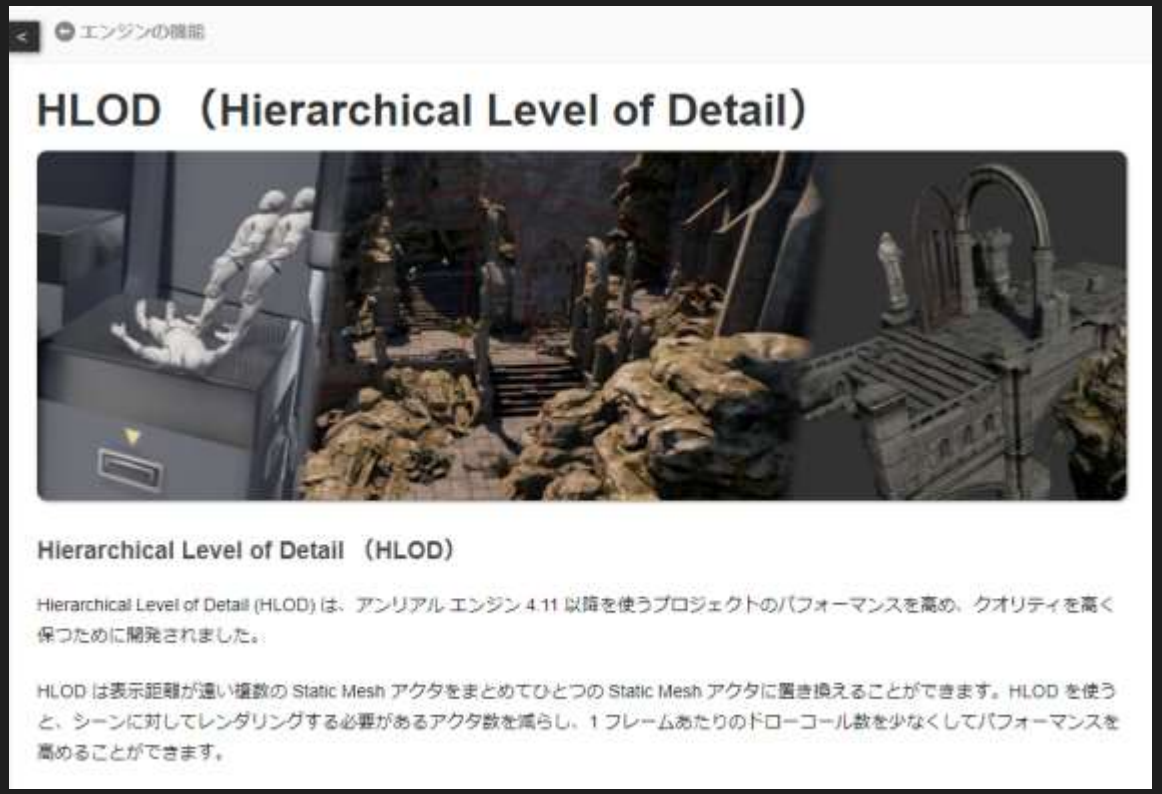
- ポリゴンが減ることでGPU負荷の削減
- Materialが減り、Draw Callを減らすことができる

注意点

- MaterialをマージしないとDraw callは減らない
- 保持すれば保持するほどメモリ量は増える

HLODの使い方

- HLODの使い方に関しては、ドキュメントをご参照ください
- http://api.unrealengine.com/JPN/Engine/HLOD/



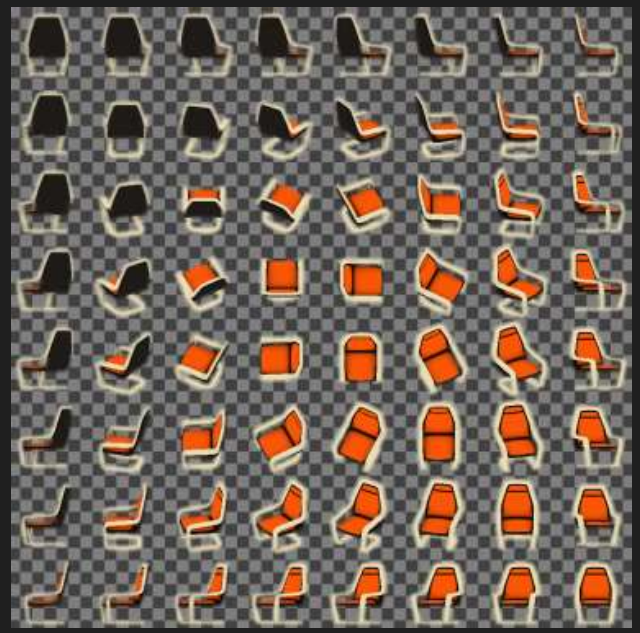
HLOD + Inpostor

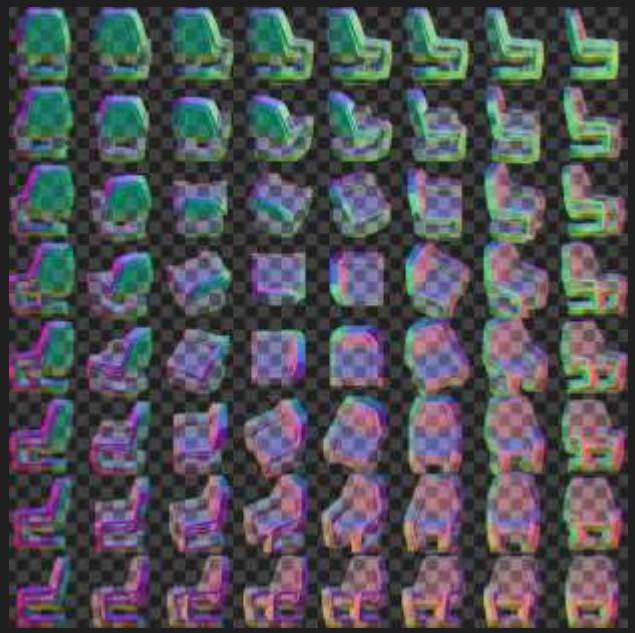
HLODでガッツリ削ると
遠景とはいえクオリティが下がってしまう


12,906 tris
Standard HLOD

Inposter

- アセットを各方向からレンダリングしたテクスチャを用意し、
- ビルボードに貼り付けることで見せかける手法





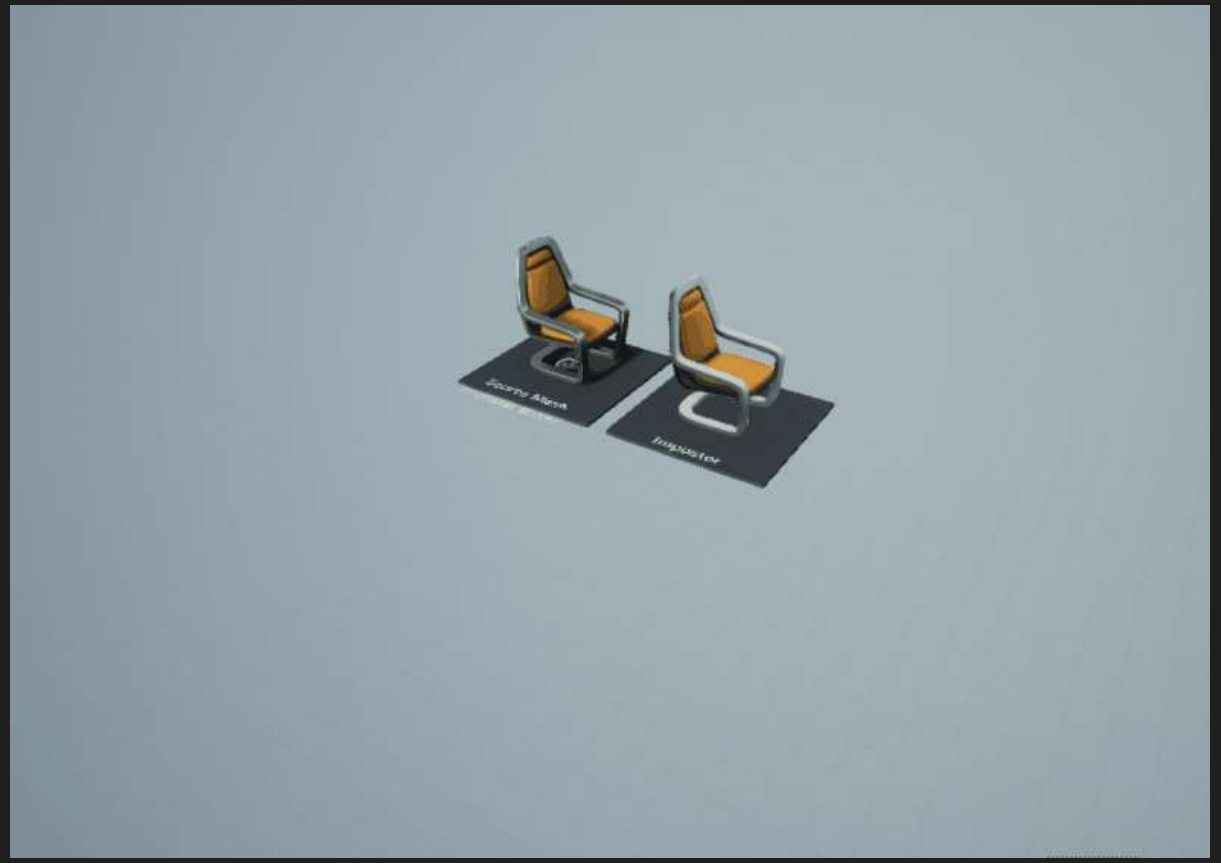
Inpostorについて

○ 利点

- ポリゴン数の大幅削減
- 負荷が高いアセットも
  一律コストで描画が可能

△ 気をつけるべきこと

- 影がでない。
- 解像度、方向数など、
  調整できるが、近いと崩れやすい。
- あくまで遠景用として

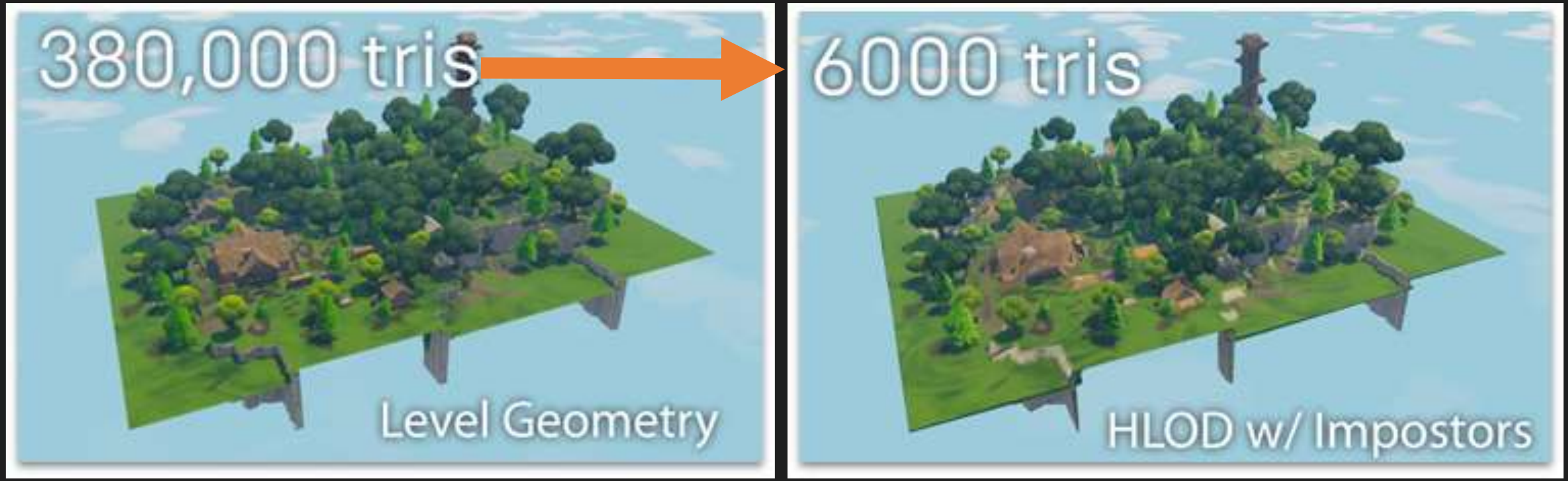


HLODとInposter組合せ

- どちらもHLODなのでDraw call数は同等
  半分のポリゴンになっていながら、見た目が綺麗

HLODとInposter組合せ

- もとのオブジェクトと比較しても、見た目を保っている



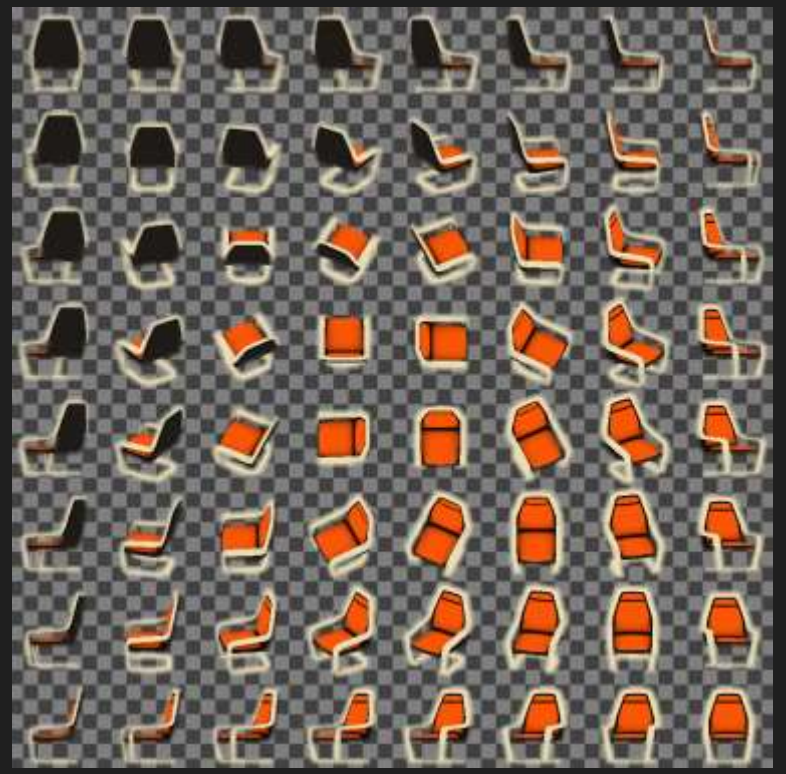

380,000 tris
Level Geometry


12,906 tris
Standard HLOD


6000 tris
HLOD w/ Impostors

ImpostorBaker

- 以前からImpostorの機能はありましたが、FortniteではPluginとして公開している、ImpostorBakerを使用
- Impostorアセットの作成が容易になり、半球分のみ生成するなどの工夫から、テクスチャの使用効率があがっています
- 詳しい使い方は付録で紹介



アセットリダクション終わり

アニメーション

Battle Royaleの特徴③

100人が同時に対戦
移動やアニメーションが同時に起こる

重要度に応じて、フレームをスキップするなどの最適化

Update Rate Optimization(URO)

Update Rate Optimization(URO)

- 遠くにいるキャラクター等、重要度の低いものに対して
  アニメーションの更新頻度を抑えることで処理を軽くする。

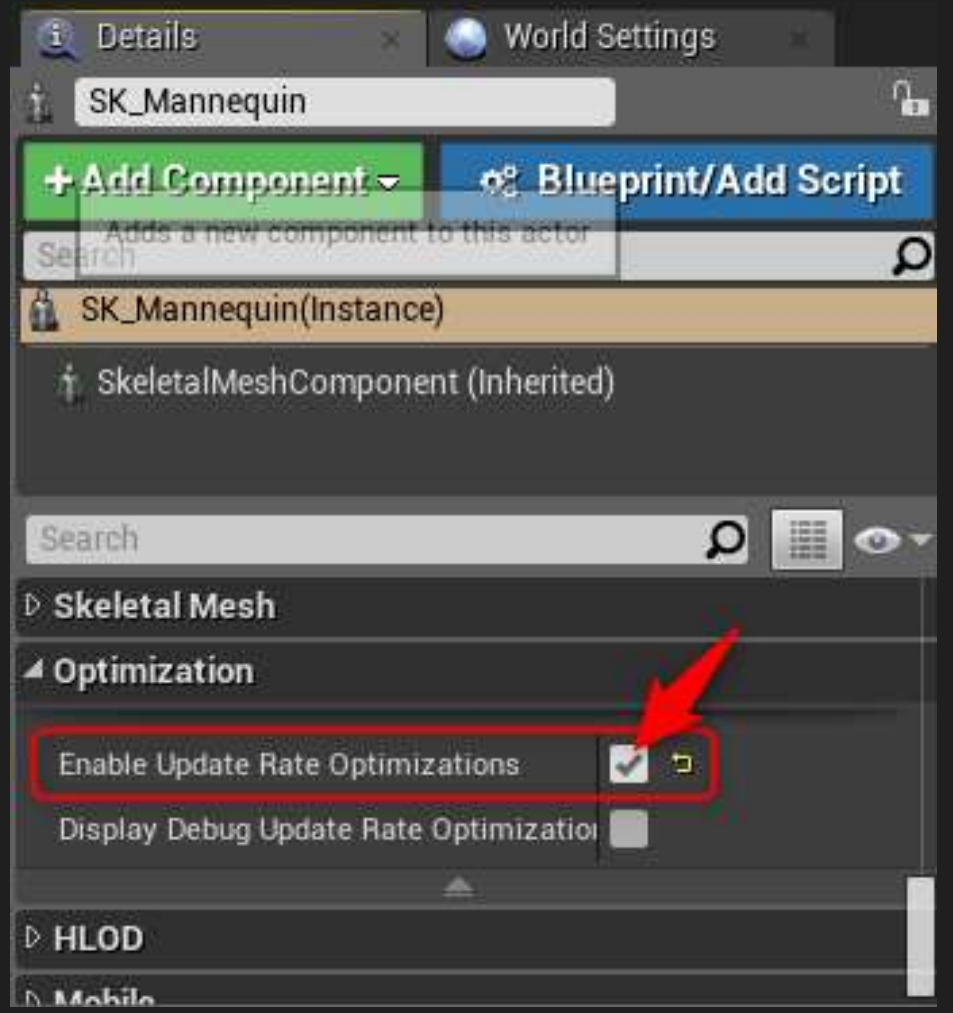
No URO

URO 5

URO 10

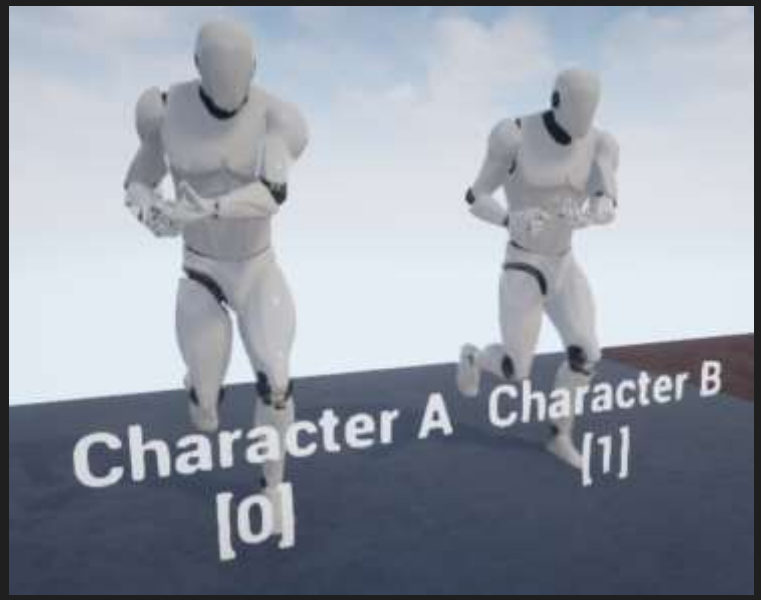
UROの設定方法

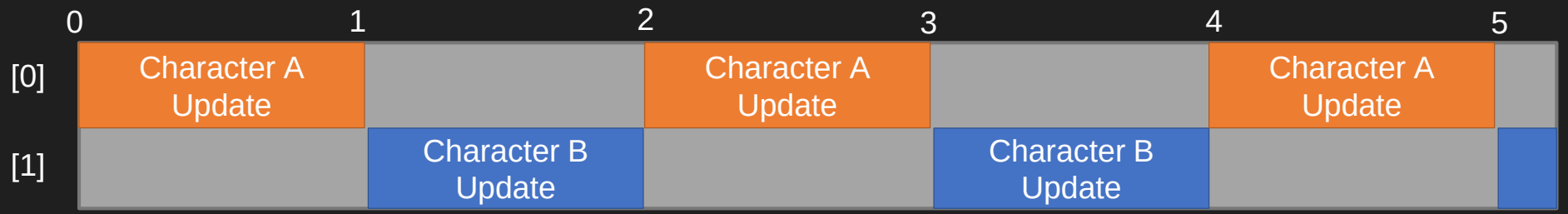
Skeletal Mesh Compornentの
Enable Update Rate Optimizationsを
有効にすることで動作可能



UROのイメージ

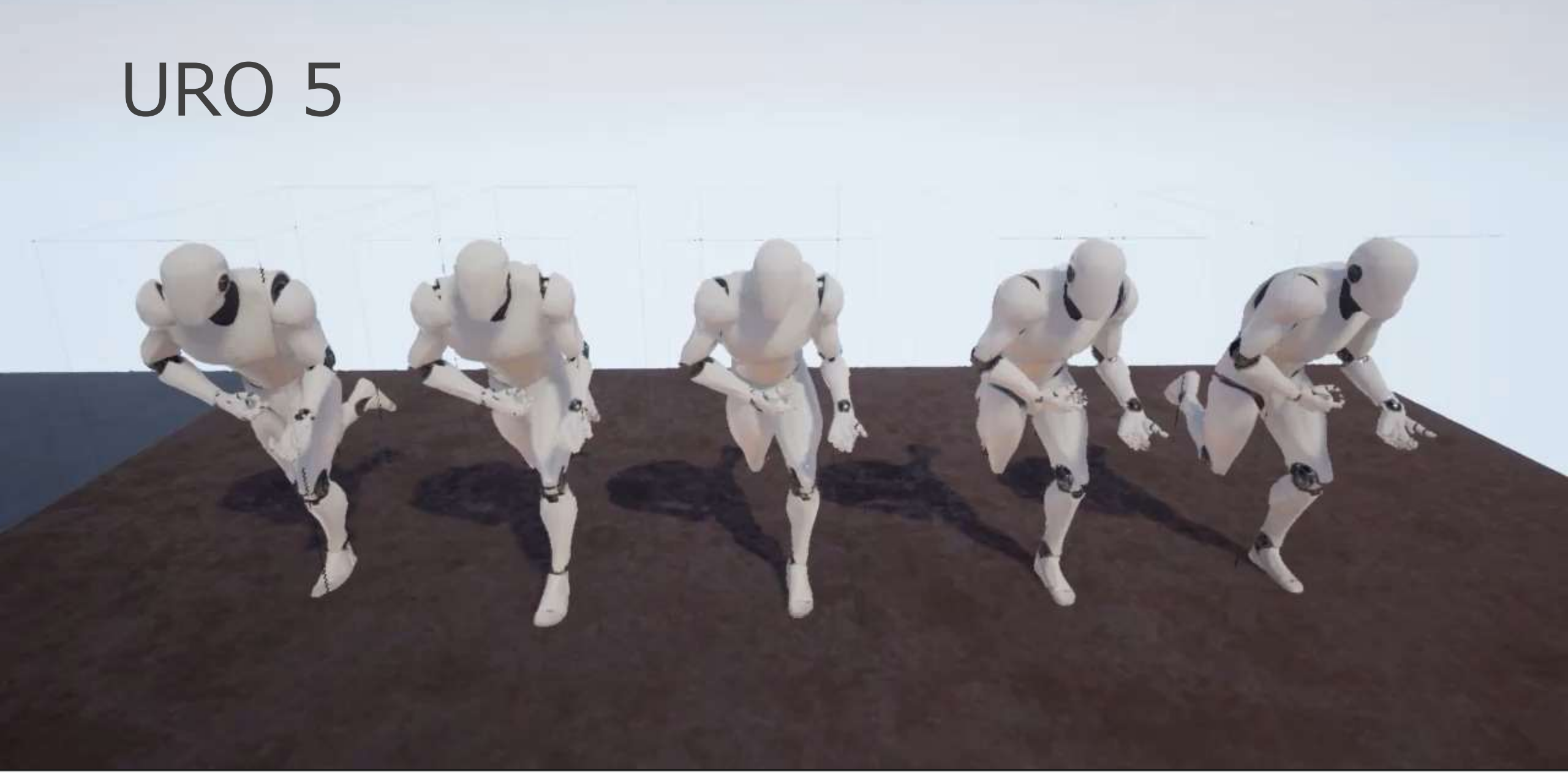
- キャラクターA、Bがいたとした場合
  - それぞれにインデックスが割り振られる
  - すべてのキャラを2フレームごと更新と設定
- 交互にアップデートするように最適化される





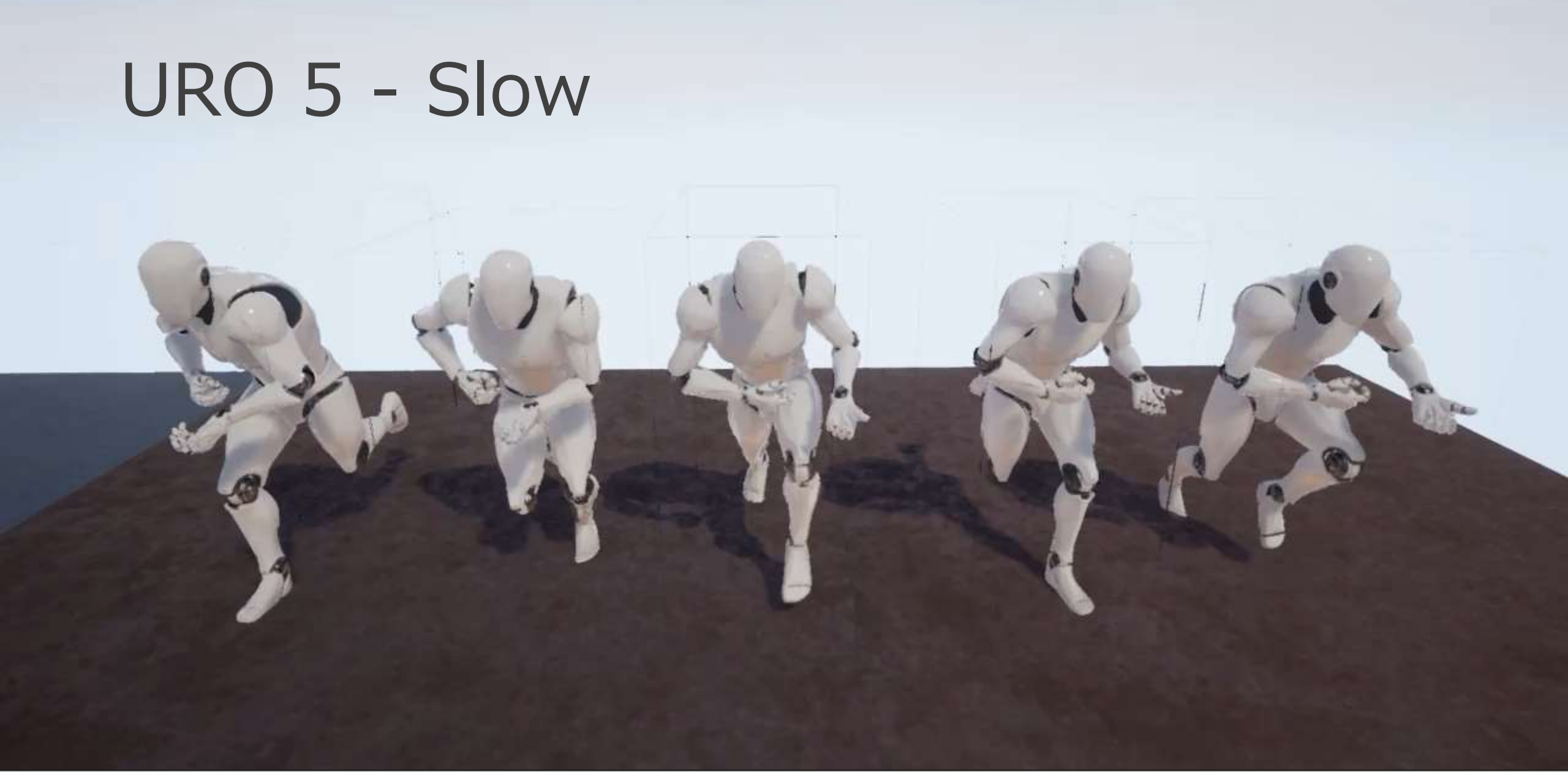
5フレームスキップしたUROの例

URO 5



ゆっくりみてみると...

URO 5 - Slow



UROのパラメーター設定

- 以下コマンドをOutputLogから入力することで検証可能

コマンド	効果
a.URO.Enable	強制的にUROをON,OFF
a.URO.ForceAnimRate	何フレーム毎にするか強制的に設定する

- 現状UIから細かい設定ができず、C++からの設定が必要

アニメーション終わり

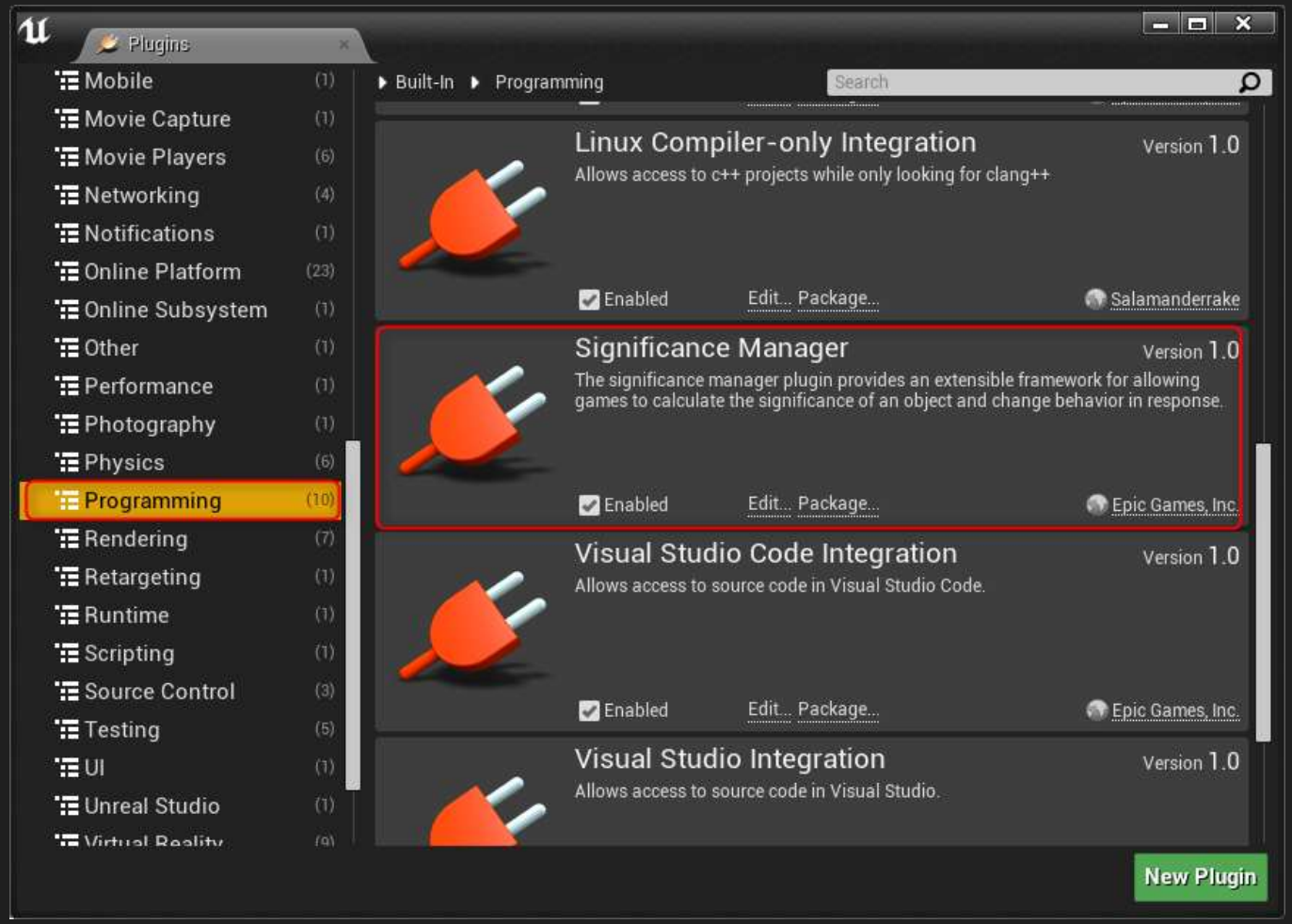
LOD、HLOD、URO、、、

紹介してきた機能のパラメーターを
どうやって切り替えているのか？

Significance Manager

Significance Manager

4.15からプラグインとして
追加された機能(クラス)

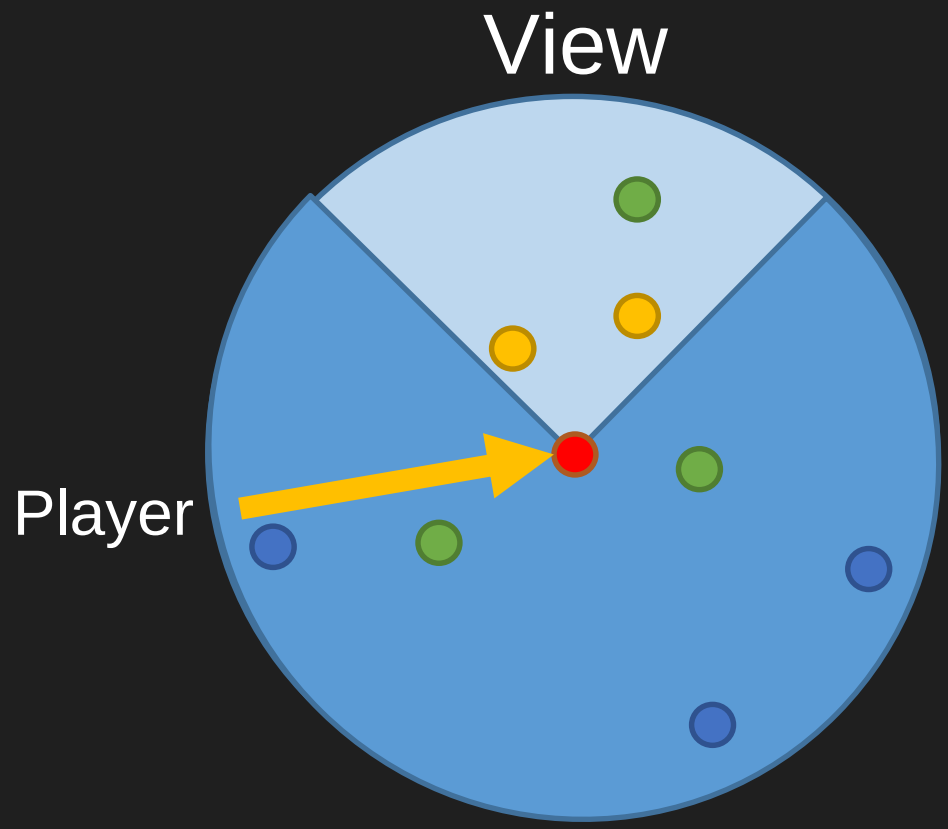


Significance Manager

ビューの表示状態をもとに、優先度(状態)を管理

- LOD
- HLOD
- アニメーションのURO rate

等々

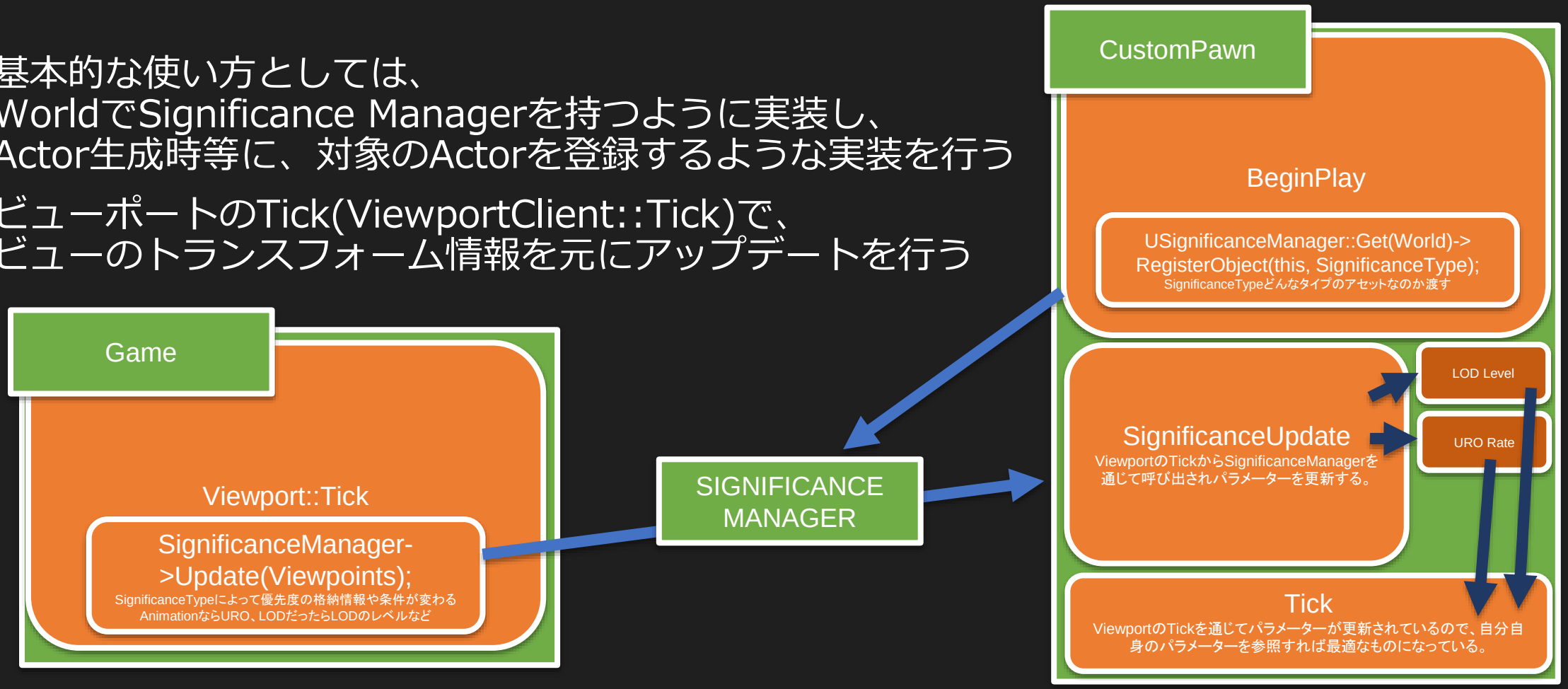


Significance Manager

UI等はないので、C++からのみアクセス可能
ドキュメントは現状ないので、
コードから理解することが必要...

Significance Manager

- 基本的な使い方としては、
  WorldでSignificance Managerを持つように実装し、
  Actor生成時等に、対象のActorを登録するような実装を行う
- ビューポートのTick(ViewportClient::Tick)で、
  ビューのトランスフォーム情報を元にアップデートを行う



Significance Manager

現状

- Significance Managerを継承して実装が必要
- ビューポートからの距離などの判断基準を自前で実装する必要がある

利点

- 一括して変更が可能なため検証がしやすい

気をつけるべきこと

- グローバルだからと、なんでも実装してしまうと肥大化する可能性

Significance Manager終わり

まとめ

今回はFortniteに関連して4.19で公開されている機能で、実際に使えそうな機能を紹介しました

お役に立てる情報があれば幸いです

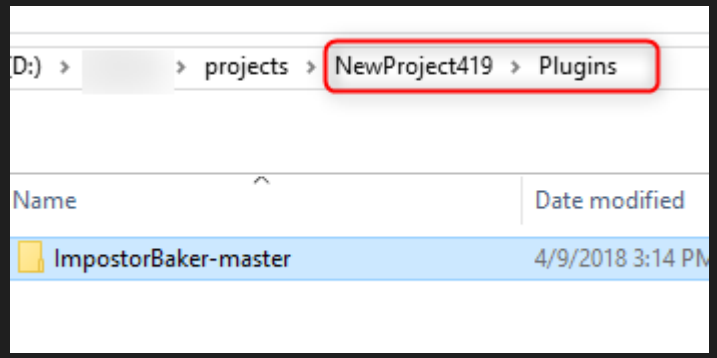

UNREAL ENGINE

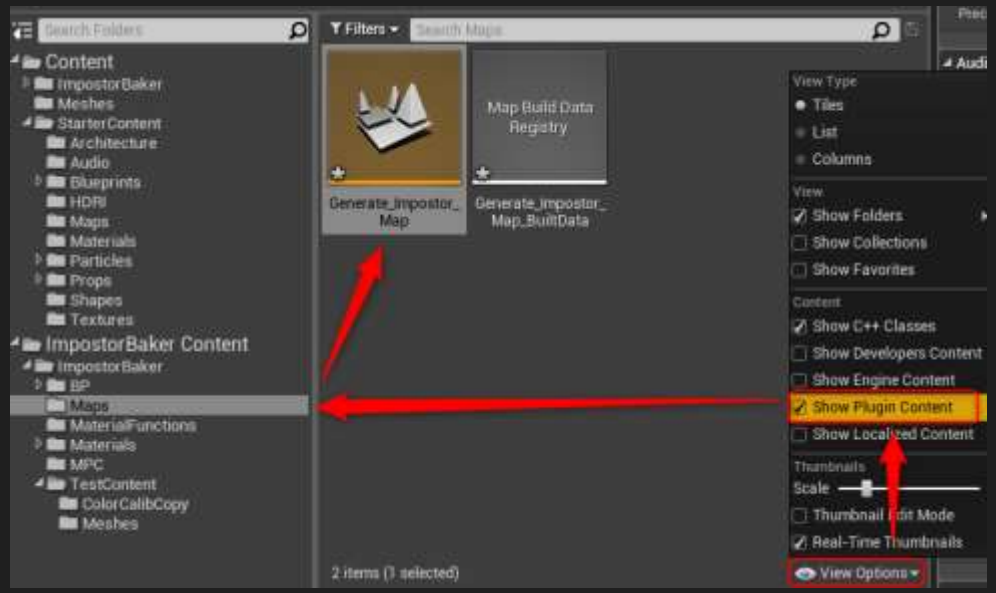
付録

ImpostorBakerの使い方

ImpostorBaker

- 準備
  - https://github.com/ictusbrucks/ImpostorBaker からプラグインをダウンロード。
  - Project以下のPluginsフォルダにダウンロードしたプラグインを追加。
  - コンテンツブラウザーからPluginコンテンツを表示する。
  - Generate_Impostor_Mapを開く



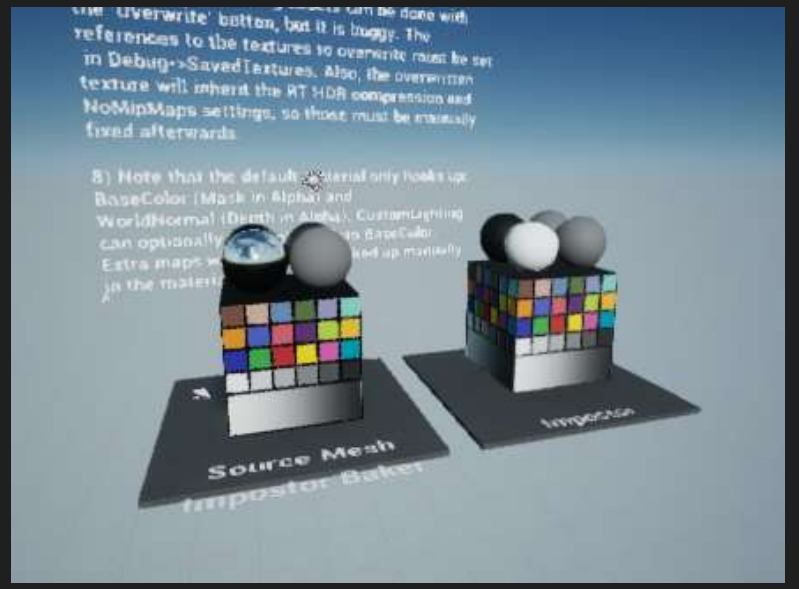


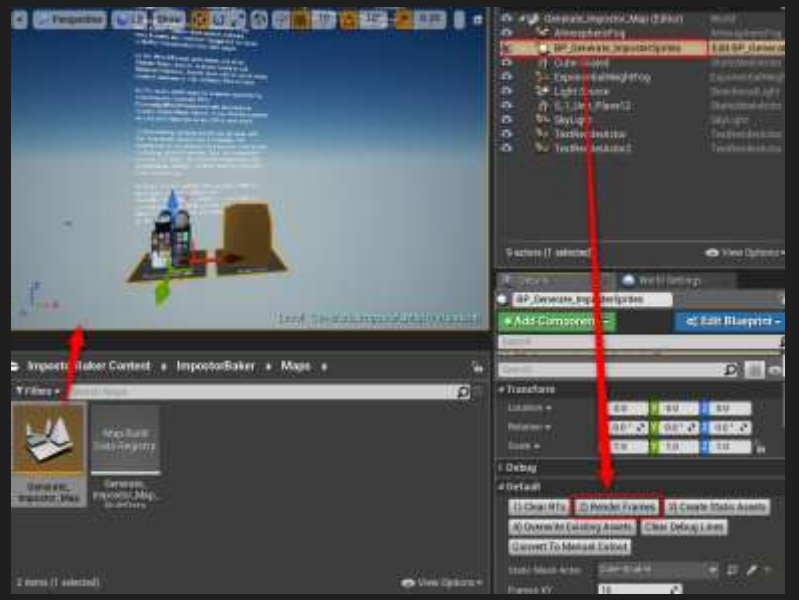
ImpostorBaker

- はじめに

Generate_Impostor_Mapを開いた後に、
Levelに配置されているBP_Generate_ImposterSpritesを選択する。

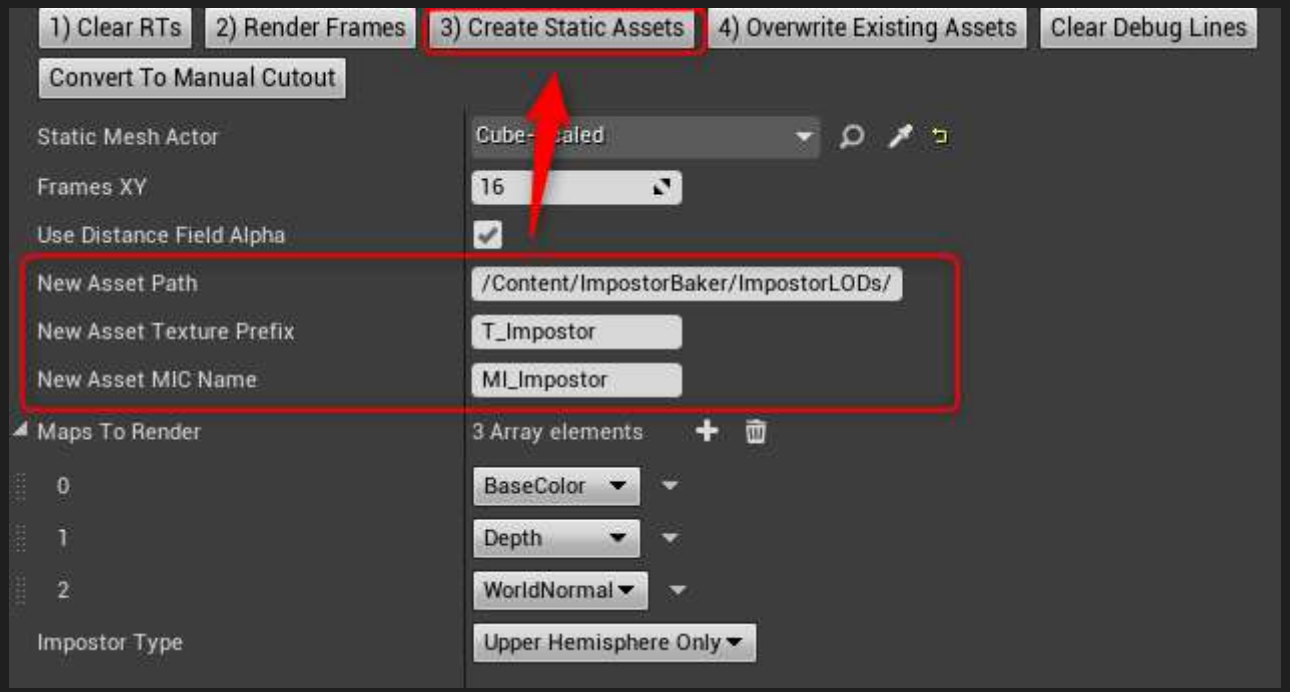
Render Framesボタンをクリックすると、
初期配置されているアセットのImpostorのプレビューができる。





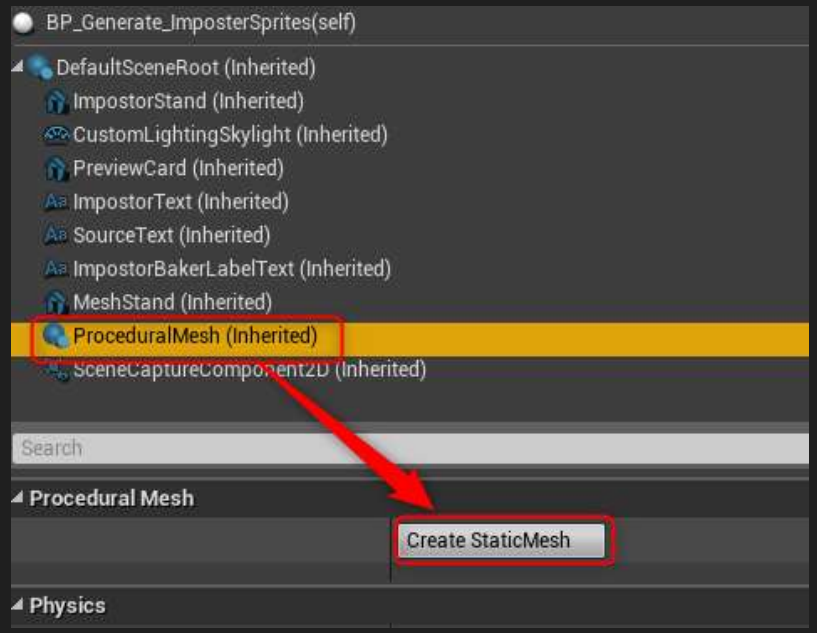
ImpostorBaker

- マテリアルの出力
  - BP_Generate_ImposterSpritesのDetailsから出力先情報を入力し3)Create Static Assetsをクリック



ImpostorBaker

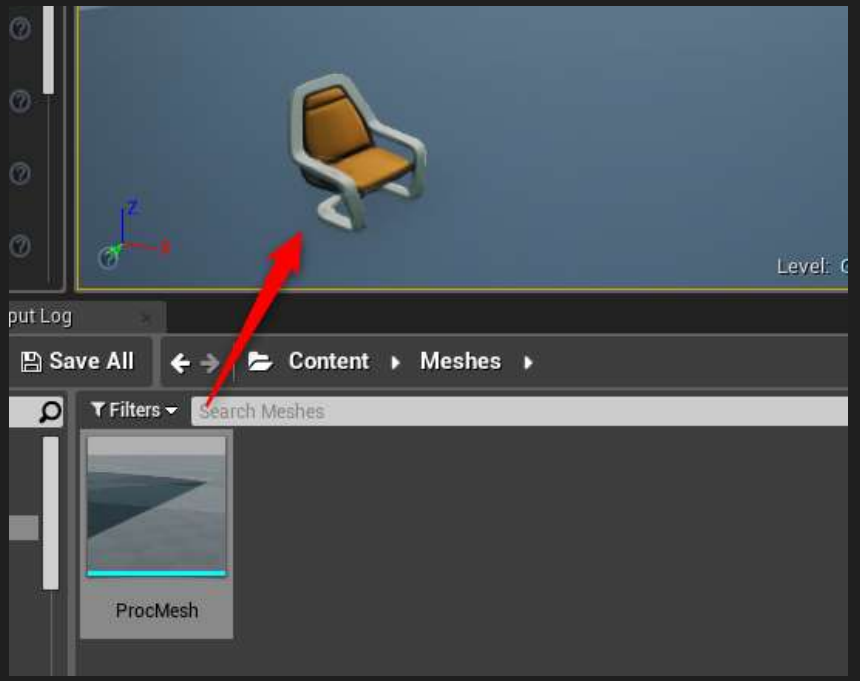
- メッシュの出力
  - BP_Generate_ImposterSpritesのコンポーネントからProceduralMeshを選択する。
    一覧からProceduralMeshのCreate StaticMeshをクリックすると出力先を聞かれるので選択する。



ImpostorBaker

Inpostorのスポーン

- ProceduralMeshから生成したStatic Mesh Actorを Levelにスポーンさせれば、生成したInpostorを確認できる。

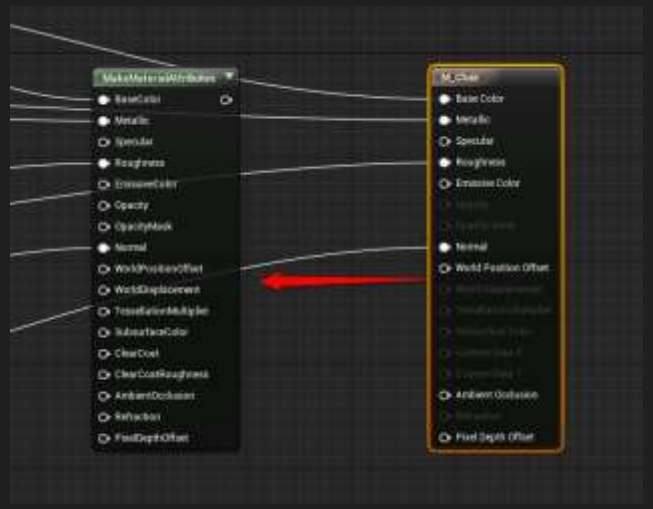


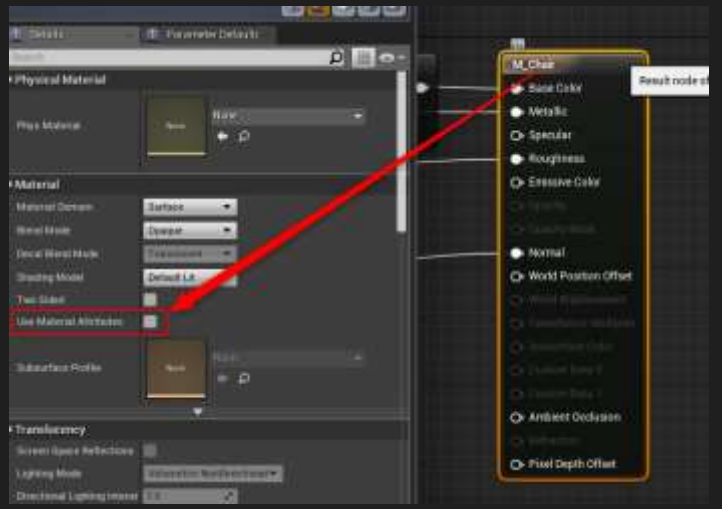
ImpostorBaker

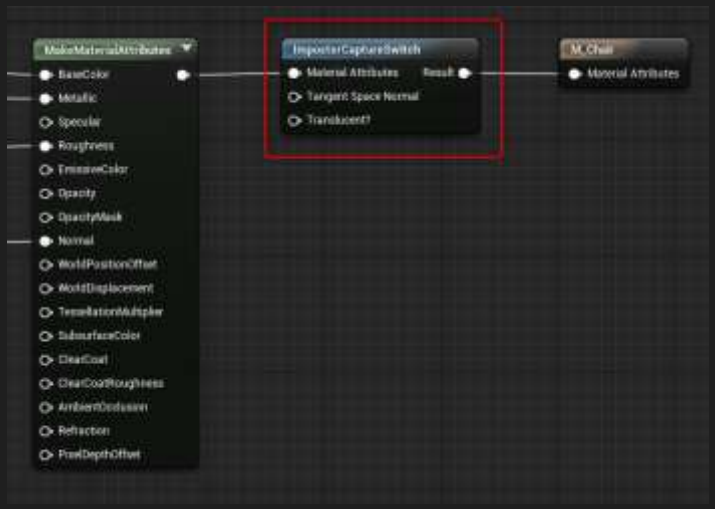
初期アセット以外のセットアップ

ImpostorBaker

- Materialのセットアップ
  - 使用しているMaterialの最後にImpostorCaptureSwitchを追加する。
    1. 通常のMaterial Nodeの接続をMakeMaterialAttributesに置き換る
    2. この時出力ノードをDetailsからUse Material AttributesをEnableにする
    3. ImpostorCaptureSwitchを挟み出力させる。



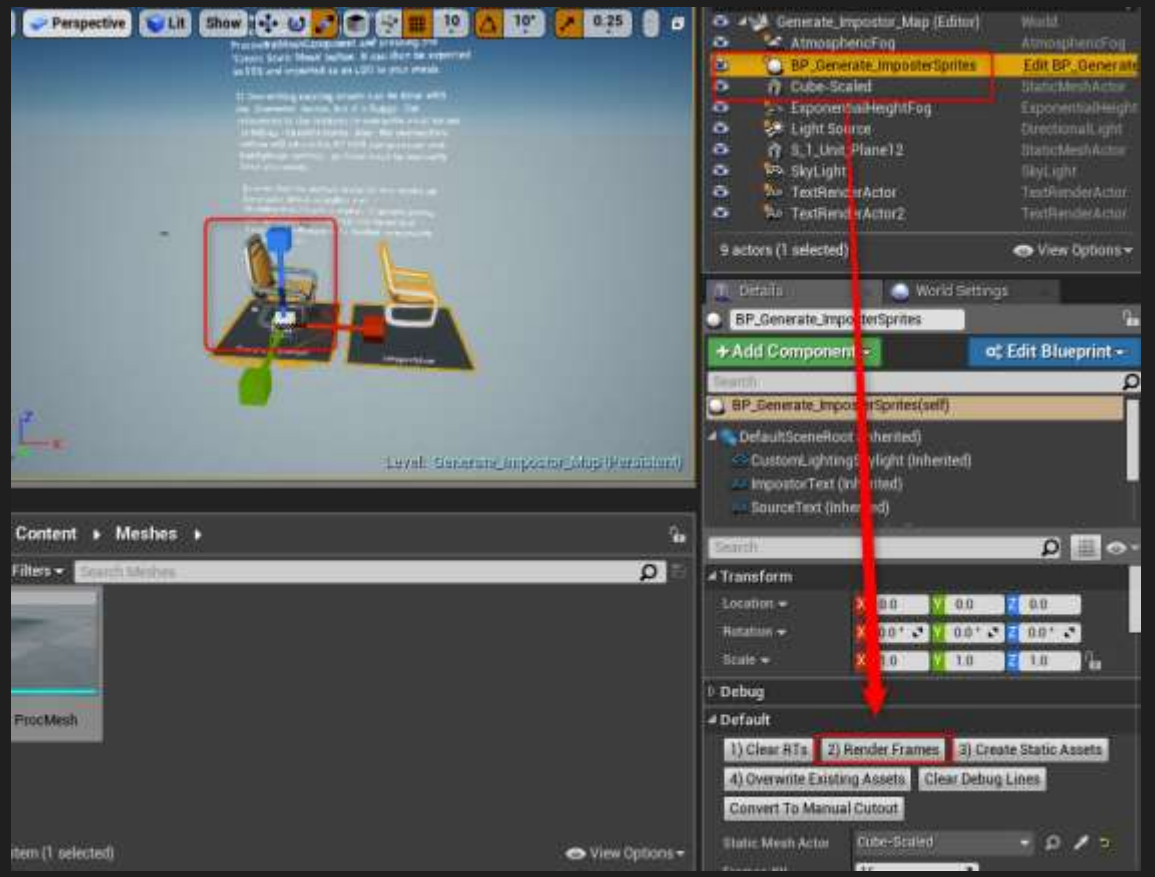




ImpostorBaker

- Static Mesh Actorの置き換え

  Materialを設定したStaticMeshをデフォルトのCube-Scaledと置き換えるような形で配置し、RenderFramesで確認する。



GDC講演Youtubeリンク

Fortnite最適化セッション

- Optimizing UE4 for Fortnite: Battle Royale - Part 1
  https://www.youtube.com/watch?v=KHWquMYtji0&index=16&list=PLZlv_N0_O1gYcueZ-mOraUuH0WufCLuto&t=0s

- Optimizing UE4 for Fortnite: Battle Royale - Part 2
  https://www.youtube.com/watch?v=1xiwJukvb60&index=15&list=PLZlv_N0_O1gYcueZ-mOraUuH0WufCLuto&t=0s

その他講演プレイリスト

- GDC 2018 | Unreal Engine
  https://www.youtube.com/playlist?list=PLZlv_N0_O1gYcueZ-mOraUuH0WufCLuto