How To Use HLODSystem

Writer Jangkyu Seo (jangkyu@unity3d.com)

Reviewed by Victor Lui (victorl@unity3d.com)

Last edit Oct 25, 2019 (@bek)

Getting the Package **2**
Prerequisites 2
Getting HLOD System 2
CLI 2
Sourcetree 3
Importing package to a Unity Project 4
Using HLOD in the Project 5
Creating the Test Scene 5
Applying HLOD to Objects 6
Component **11**
HLOD 11
Common 11
Simplifier 12
Batcher 12
Streaming 13
TerrainHLOD 13
Common 13
Simplifier 13
Material 14
Streaming 14
Shared 14
Simplifier 14
Streaming 15

Getting the Package

Prerequisites

- Git Client
- Unity 2019.3+ (2019.2 works too, but 2019.3+ is recommended)
- ~~Unity Technologies GitHub Account~~

Getting HLOD System

HLOD System is provided as an individual package. Currently it is available only on GitHub. In later stages of development, it will be available through Unity Package Manager.

HLOD System GitHub Repo URL is https://github.com/Unity-Technologies/HLODSystem

Follow through to get the package to your local PC and work with it.

CLI

Step 1. Run one of the following commands to clone the repo:

```
$ git clone https://github.com/Unity-Technologies/HLODSystem.git
```

or

```
$ git clone git@github.com:Unity-Technologies/HLODSystem.git
```

```
user@DESKTOP /Dev$ git clone https://github.com/Unity-Technologies/HLODSystem.git
Cloning into 'HLODSystem'...
remote: Enumerating objects: 150, done.
remote: Counting objects: 100% (150/150), done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 4179 (delta 76), reused 80 (delta 39), pack-reused 4029
Receiving objects: 100% (4179/4179), 139.22 MiB | 16.71 MiB/s, done.
Resolving deltas: 100% (2684/2684), done.
user@DESKTOP /MobileOpenWorldSample
```

Step 2. Next, change directory to the root directory of HLOD, and pull the dependencies, which are included into project as Git Submodules. They are ConditionalCompilationUtility and UnityMeshSimplifier:

```
$ cd HLODSystem
```

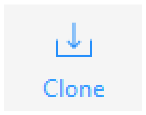
$ git submodule update --init --recursive

```
user@DESKTOP /Dev$ cd HLODSystem
user@DESKTOP /Dev/HLODSystem$ git submodule update --init --recursive
Submodule 'com.unity.hlod/Package/ConditionalCompilationUtility'
(https://github.com/Unity-Technologies/ConditionalCompilationUtility.git) registered for path
'com.unity.hlod/Package/ConditionalCompilationUtility'
...
user@DESKTOP /MobileOpenWorldSample
```

Sourcetree

For this particular example, we used Sourcetree, but the steps should be fairly similar for the GUI Client of your choice.

Step 1. Open Sourcetree, and click Clone Button:



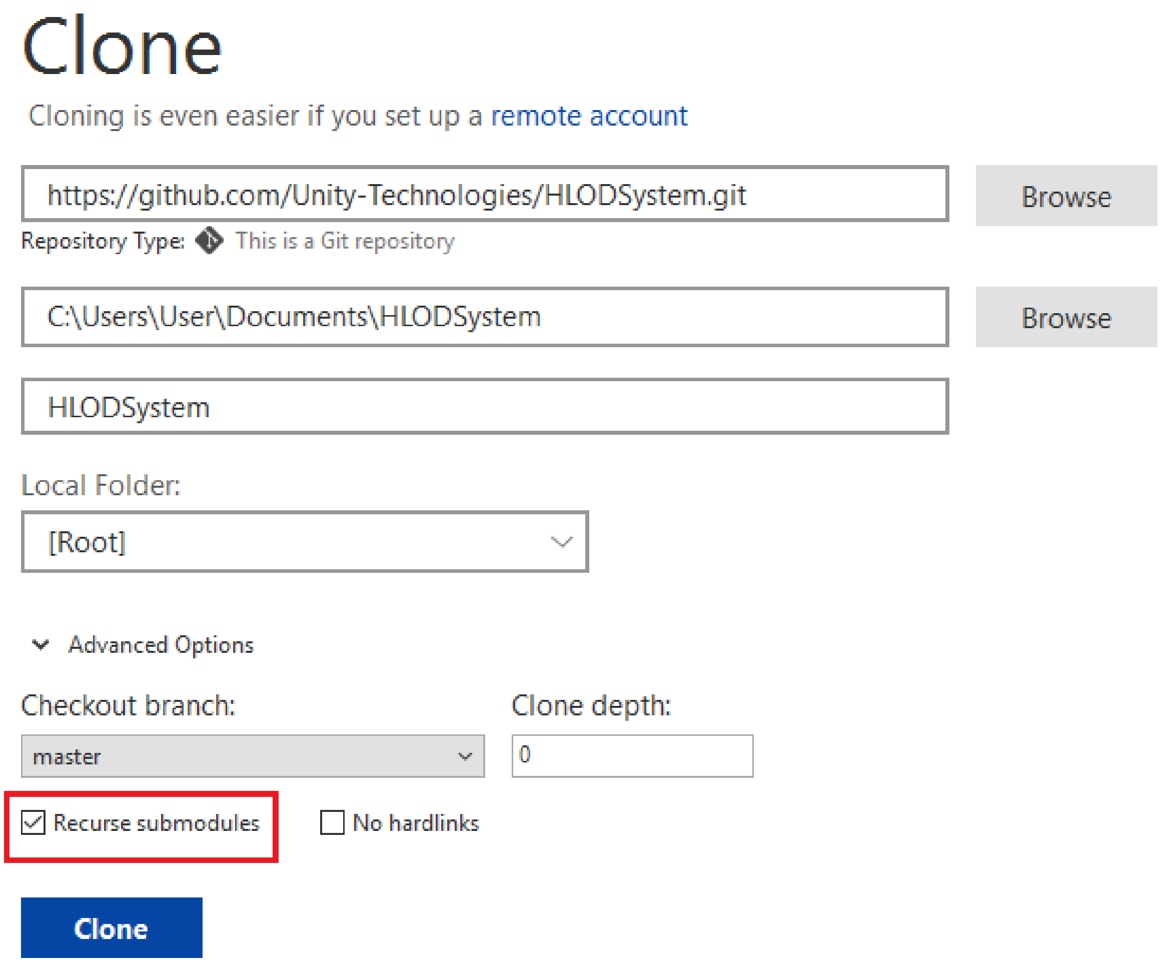
Step 2. Input the clone URL, select the destination where the repo is cloned, and wait for a couple of seconds until Sourcetree gets the repo details:

https://github.com/Unity-Technologies/HLODSystem.git

or

git@github.com:Unity-Technologies/HLODSystem.git

Step 3. Make sure to check Recurse submodules checkbox and click Clone:



Clone

Cloning is even easier if you set up a remote account

https://github.com/Unity-Technologies/HLODSystem.git

Repository Type: This is a Git repository

C:\Users\User\Documents\HLODSystem

HLODSystem

Local Folder: [Root]

Advanced Options

Checkout branch: master

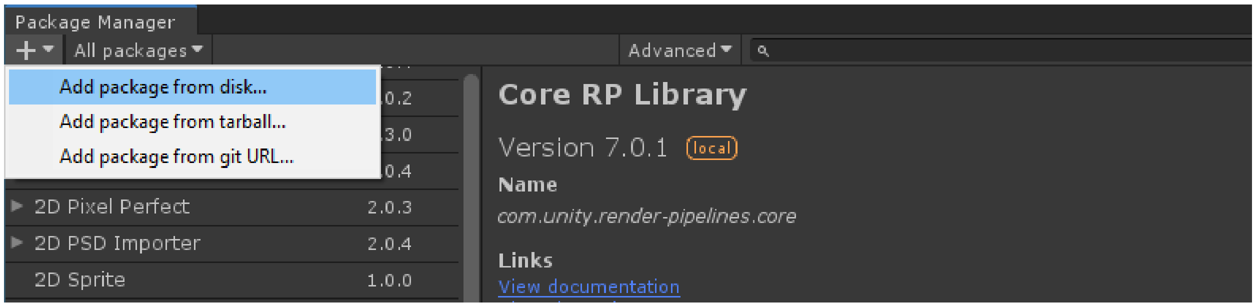
Clone depth: 0

[✓] Recurse submodules

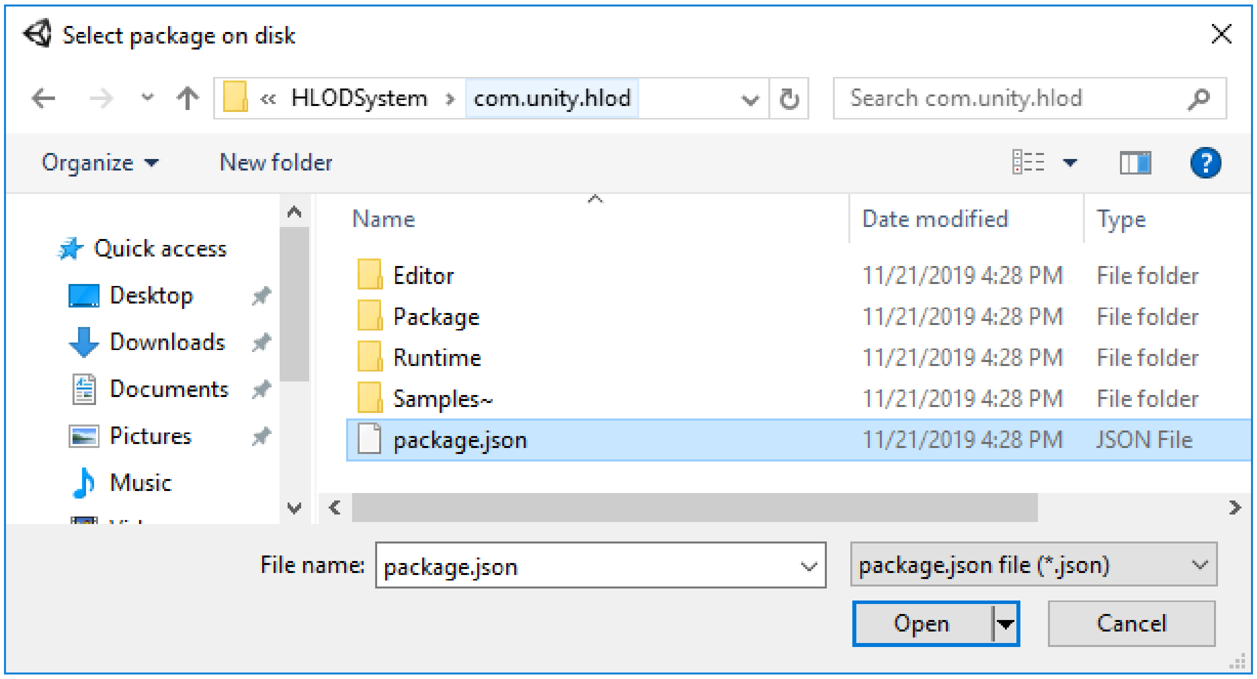
[ ] No hardlinks

Importing package to a Unity Project

Step 1. Open the Unity Project which you want to add HLOD to, and open the Package Manager Window.
Step 2. Click the + button on the top-left corner, and select Add package from disk… menu:



Step 3. Browse to the location where you cloned HLODSystem repo, open com.unity.hlod folder and select package.json file. Click Open.



Name	Date modified	Type
Editor	11/21/2019 4:28 PM	File folder
Package	11/21/2019 4:28 PM	File folder
Runtime	11/21/2019 4:28 PM	File folder
Samples~	11/21/2019 4:28 PM	File folder
package.json	11/21/2019 4:28 PM	JSON File

File name: package.json

package.json file (*.json)

The Editor will import and add HLOD System to the project. Now the project is ready to use it.

Using HLOD in the Project

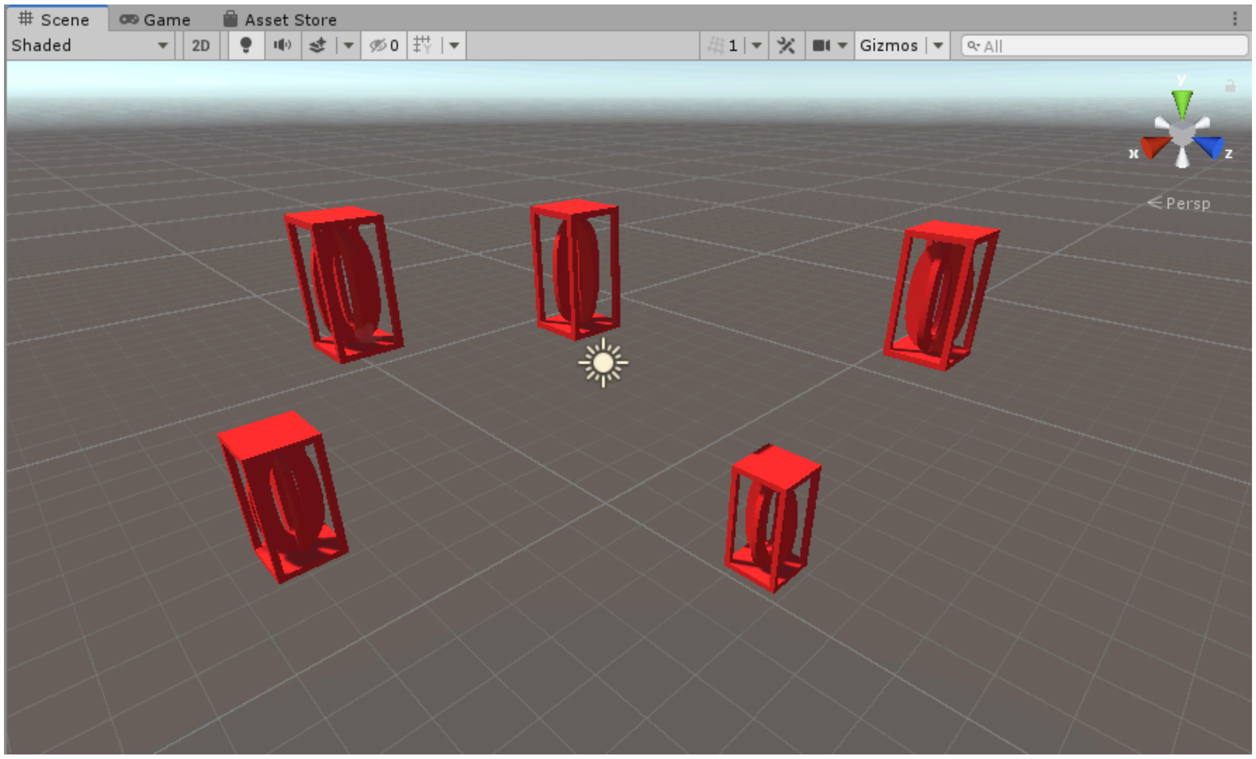
Creating the Test Scene

You can follow through the steps and create the test scene with necessary object to use with HLOD, or you can download the Test Scene created in advance and just apply HLOD.

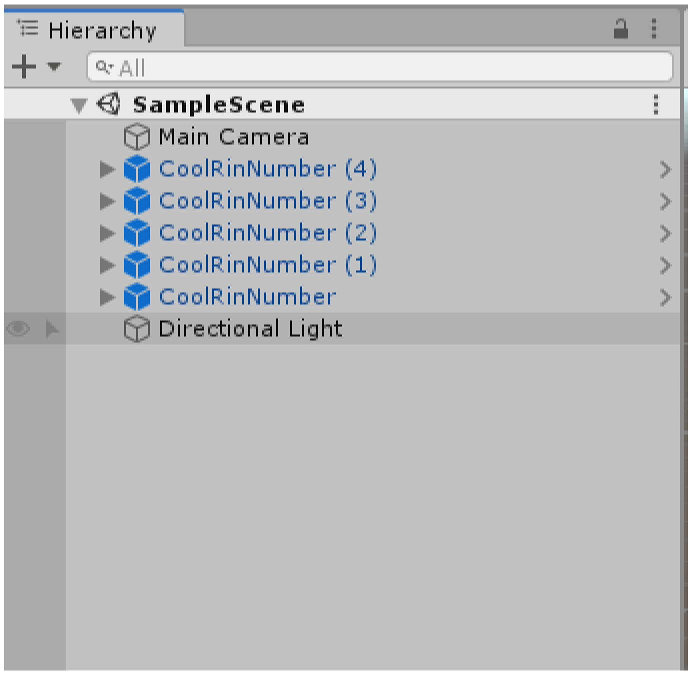
Step 1. Create a new scene and place objects in it. Let's say, 5 of them. The objects must be:

- Static
- With albedo material
- Add some more features

HLOD does not work with objects that are not static, have animated mesh, or are destroyed during game-play.



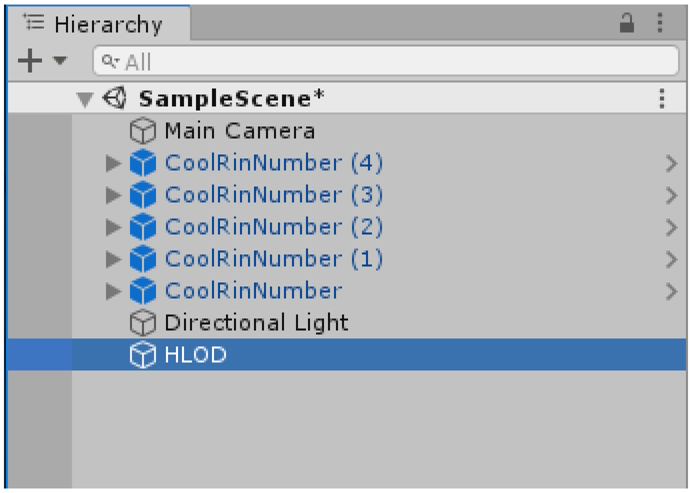
This is how Hierarchy Window looks like after we added our Game Objects:



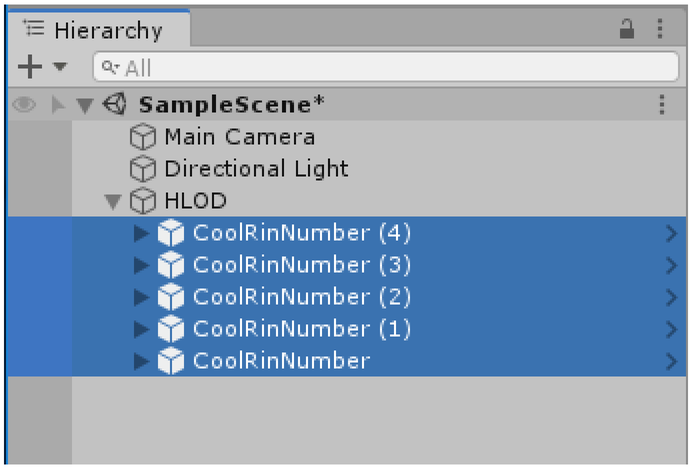
SampleScene
Main Camera
CoolRinNumber (4)
CoolRinNumber (3)
CoolRinNumber (2)
CoolRinNumber (1)
CoolRinNumber
Directional Light

Applying HLOD to Objects

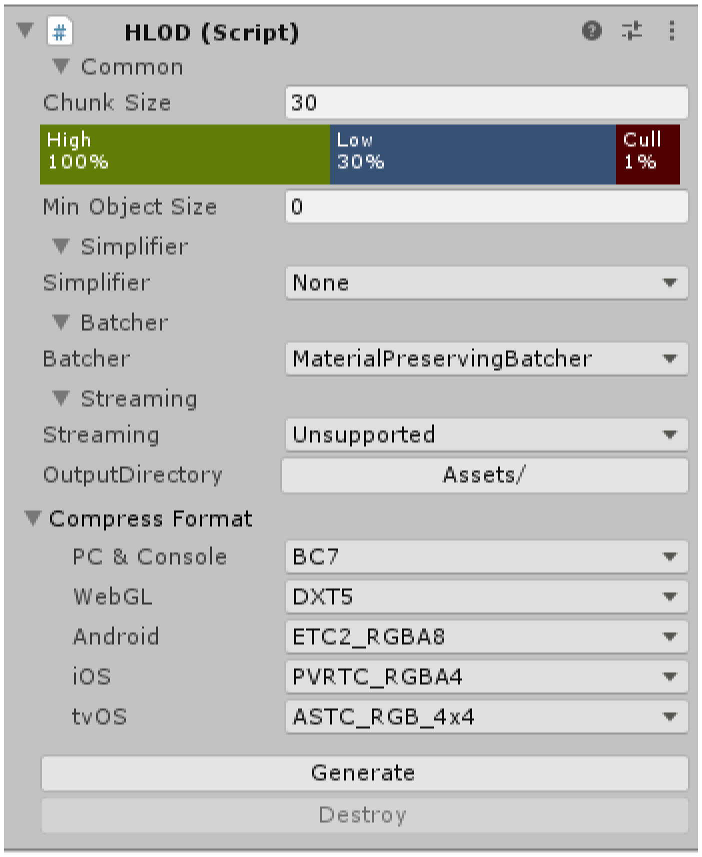
Step 1. Create an empty Game Object and name it HLOD or anything else you would like it to be:



Step 2. Select your game objects and make them the children of HLOD Game Object:



Step 3. Select HLOD Game Object and add HLOD Component to it in Inspector, then click Generate:


HLOD (Script)

Common

Chunk Size: 30

High 100% | Low 30% | Cull 1%

Min Object Size: 0

Simplifier

Simplifier: None

Batcher

Batcher: MaterialPreservingBatcher

Streaming

Streaming: Unsupported

OutputDirectory: Assets/

Compress Format

PC & Console: BC7

WebGL: DXT5

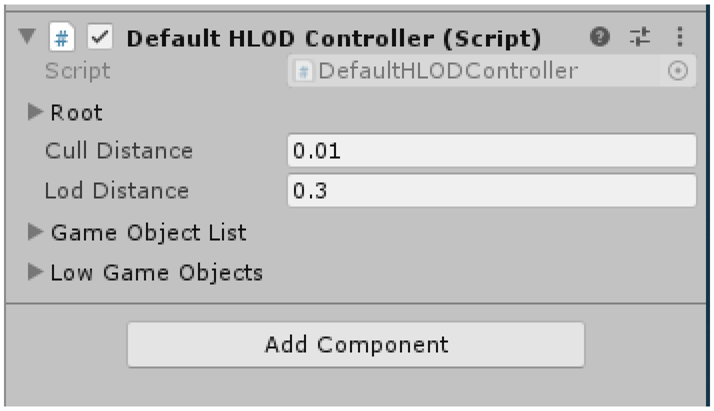
Android: ETC2_RGBA8

iOS: PVRTC_RGBA4

tvOS: ASTC_RGB_4x4

For a detailed explanation of the settings of HLOD Component, see this part of the document.

A Default HLOD Controller Component will be added automatically:


Default HLOD Controller (Script)

Script: DefaultHLODController

Root

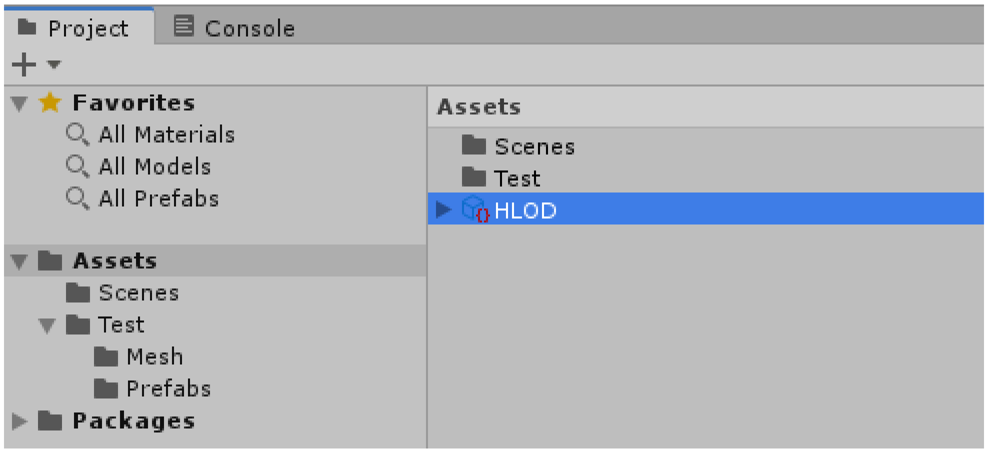
Cull Distance: 0.01

Lod Distance: 0.3

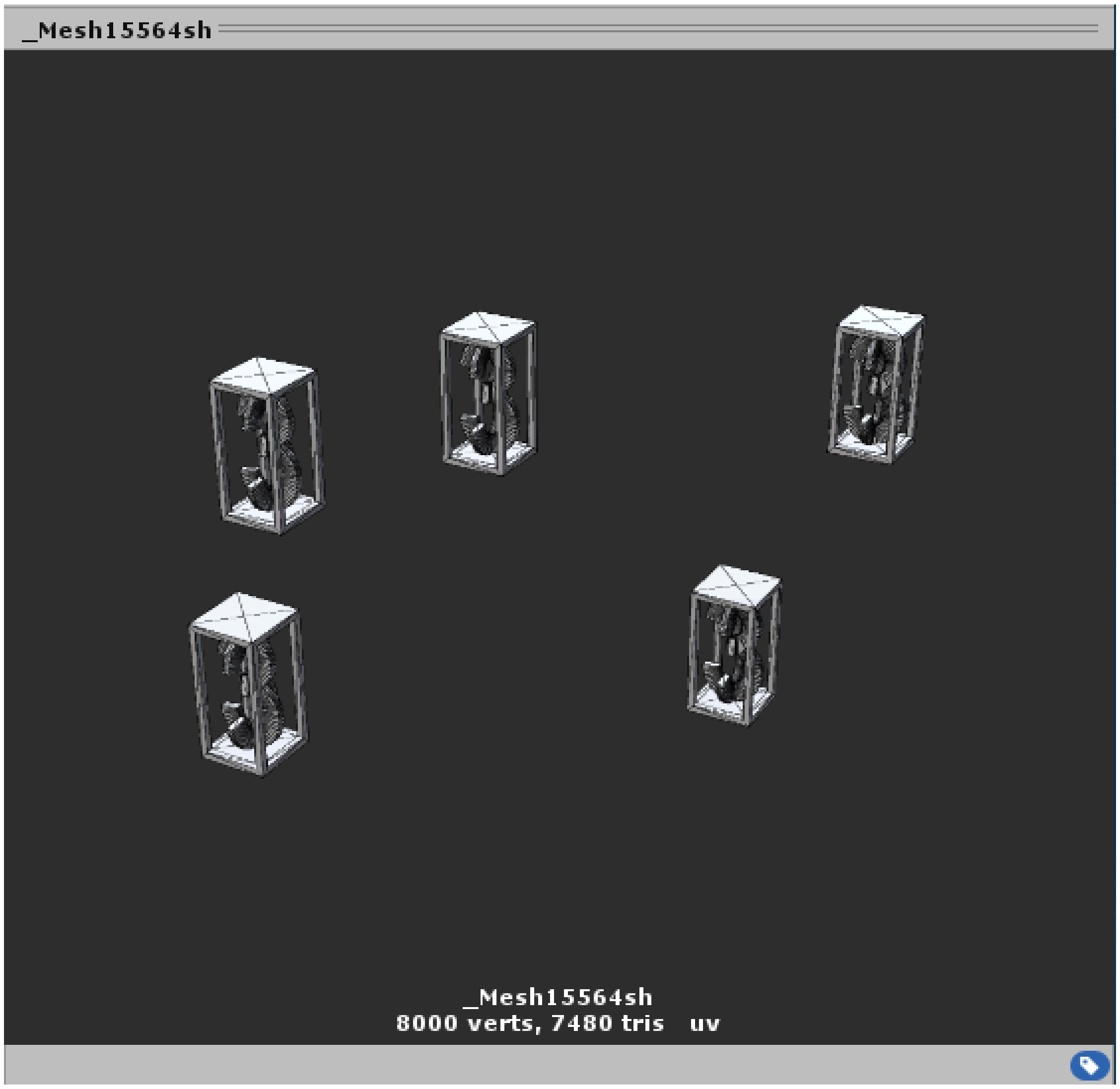
Game Object List

Low Game Objects

The HLOD Data Structure will be created in the path which OutputDirectory parameter of HLOD Component points to:



If you expand HLOD Data Structure and click the mesh component of it, you can see how Game Object Meshes are combined into a single mesh:



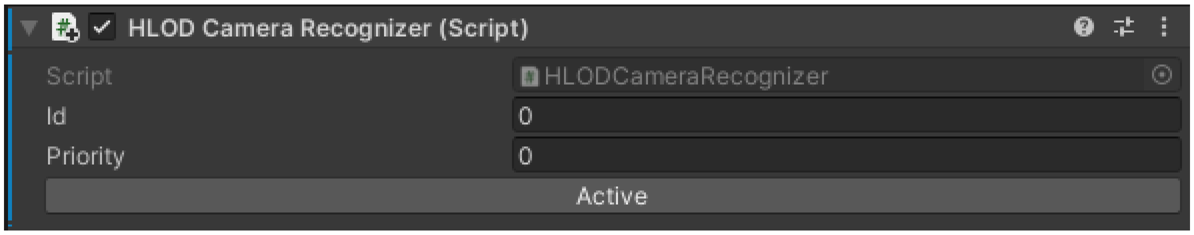
```
_Mesh15564sh
8000 verts, 7480 tris   uv
```

Step 4. Click Destroy button if you want to destroy and/or re-generate HLOD for a given Game Object.

Setting up Camera HLOD visibility.

When entering Play mode, none of the HLOD objects will be visible by default. This is because the Cameras used to render the scene must be registered with the HLOD system to allow it to calculate the HLOD meshes that need to be enabled as the Camera moves through the scene.

Step 1. Add an **HLOD Camera Recognizer** component to each Camera in the scene that is supposed to render HLOD controlled geometry.


HLOD Camera Recognizer (Script)

Script	HLODCameraRecognizer
Id	0
Priority	0

Component

HLODSystem provides 2 components used to build 2 types of HLOD data-structure:

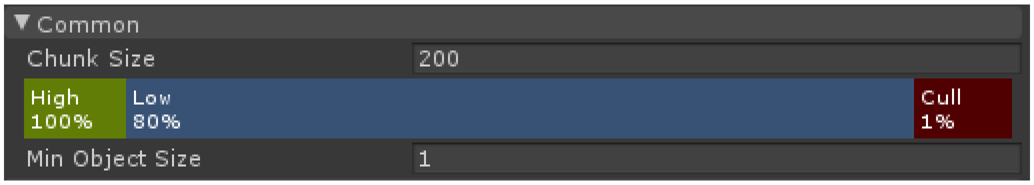
- **HLOD**: This component is used to generate HLOD data-structure for static objects.
- **TerrainHLOD**: This component is creating HLOD meshes from the Terrain data.

HLOD

The **HLOD** component generates HLOD data-structure for static objects.

Common

Typical setting related to HLOD.



▼ Common

Chunk Size 200

High 100% Low 80% Cull 1%

Min Object Size 1

Chunk Size: For our hierarchy with HLOD component, we define a number of detail levels where each level represents one way to group the meshes into a number of merged meshes. On the top level, all meshes are merged together. On the next level, we partition the meshes into 4 merged mesh (we do not partition along height axis, but only across the horizontal plane). In this way, the HLOD system builds a quadtree data structure.

The “Chunk Size” setting sets the size of the “terminal node” of the HLOD quadtree. Nodes are split into quadtrees until they are smaller than this value at full size.

High/Low/Cull: This setting defines a function that categorizes a mesh into 3 categories: *High*, *Low*, and *Cull*. Looking at the above setting as an example, if a mesh has AABB projected onto the screen occupying less than 1% of the screen, it is categorized as “Cull”. If a mesh has AABB projected onto the screen occupying more than 80% of the screen, it is categorized as “High”.

When rendering HLOD mesh hierarchy, we traverse the HLOD quadtree. If the root HLOD mesh is “Low”, we render just the combined HLOD mesh. If the root HLOD mesh is “Cull”, we render nothing. If the root HLOD mesh is “High”, we look at the children of the HLOD root node and decide how to render each children recursively in the same fashion.

Min Object Size: Specifies the minimal size for a mesh to be included in the HLOD data-structure. If the size of a mesh is greater than the set value, it is included in the HLOD system. If a mesh is too small, it is excluded from the HLOD mesh.

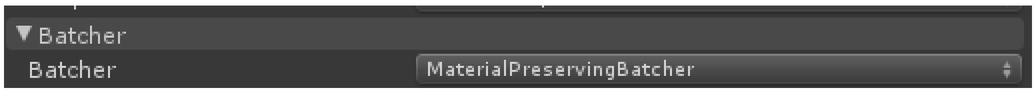
Simplifier

This is a setting shared with TerrainHLOD. Please see Shared.Simplifier section.

Batcher

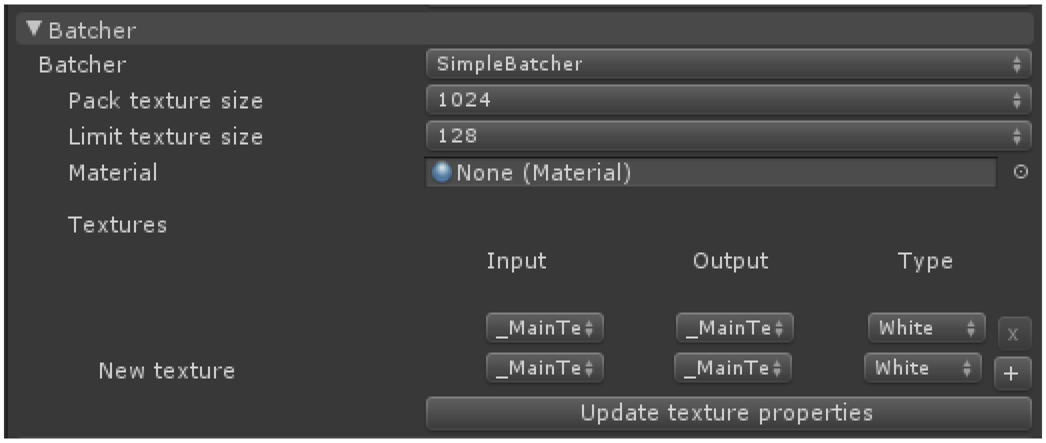
This setting determines how the meshes are combined. We provide two options here:

MaterialPreservingBatcher


▼ Batcher
Batcher: MaterialPreservingBatcher

HLOD meshes are created by grouping meshes with the same material. We use the existing material as it is.

SimpleBatcher


▼ Batcher
Batcher: SimpleBatcher
Pack texture size: 1024
Limit texture size: 128
Material: None (Material)
Textures

	Input	Output	Type
	_MainTe	_MainTe	White
New texture	_MainTe	_MainTe	White

Update texture properties

Even if the material of adjacent meshes is different, we always merge the meshes and create a new material for the merged mesh. To do this properly, we need to combine the textures referenced by different materials into textures referenced by the new material, and accordingly perform UV-remapping when building the merged mesh. We have a number of settings that specifies how this process should be performed:

Pack texture size: Sets the size of the generated texture atlas.
Limit texture size: Sets the maximum area each source texture occupies. If (because of this setting) there is unused space after all source textures are copied to the generated texture atlas, HLOD system will try to reduce the size of the generated texture atlas.
Material: Sets the material for the merged mesh. If not set, we'll use *Standard Shader*.

Textures: An entry X-Y here means we find all textures referenced as property X from materials used by meshes in the HLOD hierarchy, merge them together into a combined texture atlas and set this texture atlas as parameter Y in the new generated material.

Note that this means HLOD system would only work properly if all the materials used by the meshes in the HLOD hierarchy have a similar texture input naming convention.

Update texture properties: The **Textures** setting tries to collect appropriate material properties for the GUI drop-down, but sometimes this does not work properly. Press this button if some material property you want to use for the **Textures** setting is missing.

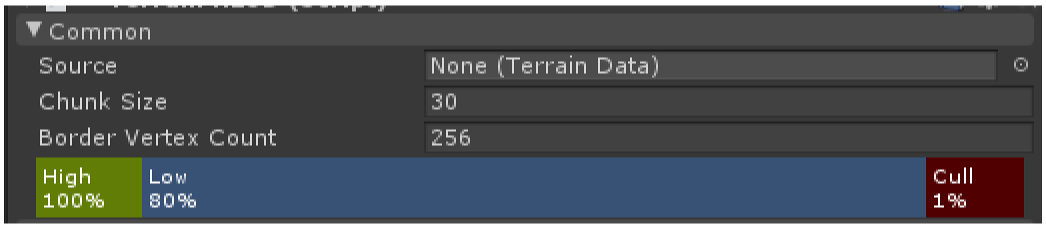
Streaming

This part is shared with TerrainHLOD. Please see Shared.Streaming section.

TerrainHLOD

TerrainHLOD takes a Terrain as input and converts it to a HLOD Mesh.

Common



▼ Common

Source	None (Terrain Data)
Chunk Size	30
Border Vertex Count	256

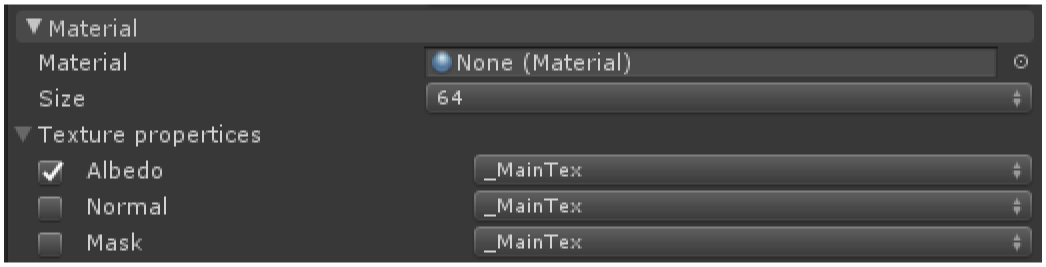
Source: Set the source TerrainData to generate the HLOD.
Chunk Size: Sets the size of the terminal node in HLOD. Nodes are split into quadtrees until they are smaller than this value at full size.
Border Vertex Count: For each side of a terminal node terrain patch, we allocate this number of regularly spaced vertices. As we combine terrain patch for lower LOD levels, although we can simplify and reduce vertex count for inner vertices, we must preserve vertices at the edge to avoid seams. So for example if we have 3 LOD levels and Border Vertex Count is 256, terminal node have 256

Simplifier

This part is shared with HLOD. Please see Shared.Simplifier section.

Material

Set up the material to be used for the baked TerrainHLODMesh. Note that we bake the Splat System into single textures for our TerrainHLODMesh material.



Material
Material: None (Material)
Size: 64
Texture properties
[✓] Albedo: _MainTex
[ ] Normal: _MainTex
[ ] Mask: _MainTex

Material: Specifies the Material to be used for the TerrainHLODMesh.
Size: Specifies the size at which the texture will be baked.
Texture properties: We'll bake 3 textures for the material: Albedo, Normal, and Mask. Mask texture's R and G channel contains roughness and specular parameter. This property decides which material property these textures are set to.

Streaming

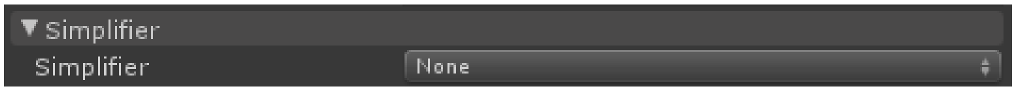
This part is shared with HLOD. Please see Shared.Streaming section.

Shared

Simplifier

Specifies how to simplify when creating HLODMesh.
Currently, there are two methods available: NotUseSimplifier and UnityMeshSimplifier.

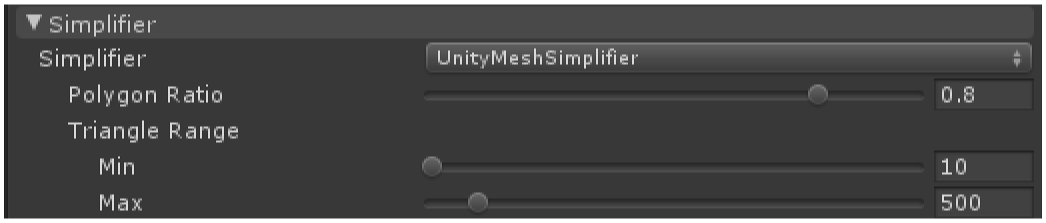
None



Simplifier
Simplifier: None

We do not simplify when creating HLODMesh.

UnityMeshSimplifier



Simplifier
Simplifier: UnityMeshSimplifier
Polygon Ratio: 0.8
Triangle Range
Min: 10
Max: 500

We simplify using UnityMeshSimplifier.

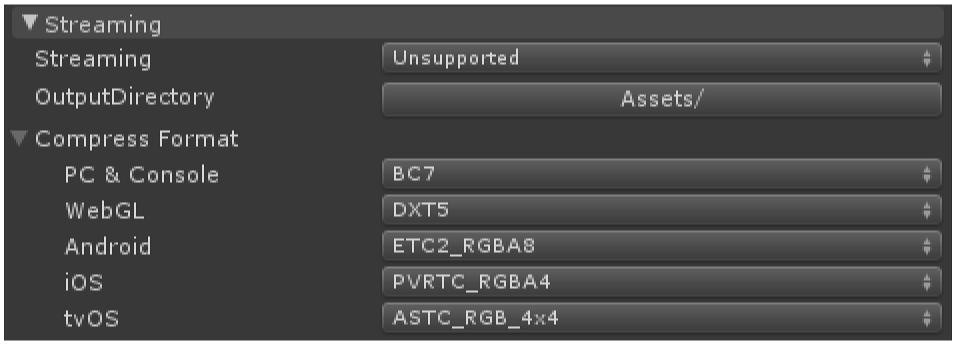
Polygon Ratio: Sets the rate at which the polygon is reduced.
Triangle Range: Sets the maximum/minimum number of polygons after simplification.

Streaming

This part is regarding how the HLOD mesh is loaded into the scene. The HLOD streaming system creates a controller component on top of the gameObject with the HLOD component. This controller is responsible for the actual HLOD mesh loading behavior. There are two modes available:

- NotSupportStreaming
- AddressableStreaming

Unsupported



▼ Streaming

Streaming	Unsupported
OutputDirectory	Assets/

▼ Compress Format

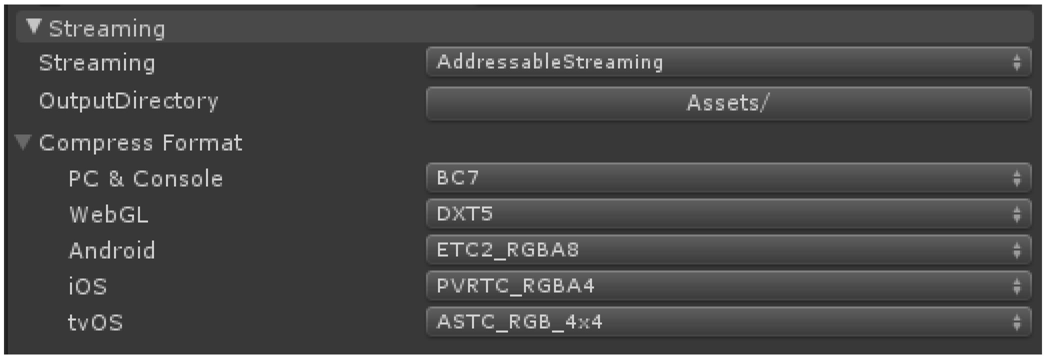
PC & Console	BC7
WebGL	DXT5
Android	ETC2_RGBA8
iOS	PVRTC_RGBA4
tvOS	ASTC_RGB_4x4

Under this mode, we do not support streaming. We add HLOD meshes at every detail level into the scene, and during gameplay we enable/disable meshes the HLOD system wants to show/hide.

OutputDirectory: Specify where the resource file will be created. The location must be under an Assets folder.

Compress Format: Specifies texture compression format by the platform. Texture compression happens when resources are imported or when the platform changes.

AddressableStreaming



▼ Streaming

Streaming	AddressableStreaming
OutputDirectory	Assets/
▼ Compress Format	
PC & Console	BC7
WebGL	DXT5
Android	ETC2_RGBA8
iOS	PVRTC_RGBA4
tvOS	ASTC_RGB_4x4

Implement

Streaming using the Addressable system. Using this mode requires user install a separate package with package ID "*com.unity.hlod.addressable*" which in turn has dependency on the Addressable system package.

Under this mode, we “appropriately” subdivide the HLOD quad-tree hierarchy into a number of separate addressable assets, each containing a group of quad-tree nodes. When a node in some group is needed, we load the addressable asset for that group into memory, then instantiate the needed node into the scene.

OutputDirectory: Specify where the resource file will be created. The location must be under an Assets folder.

Compress Format: Specifies texture compression format by platform. Texture compression happens when texture resources are imported or when the platform changes.