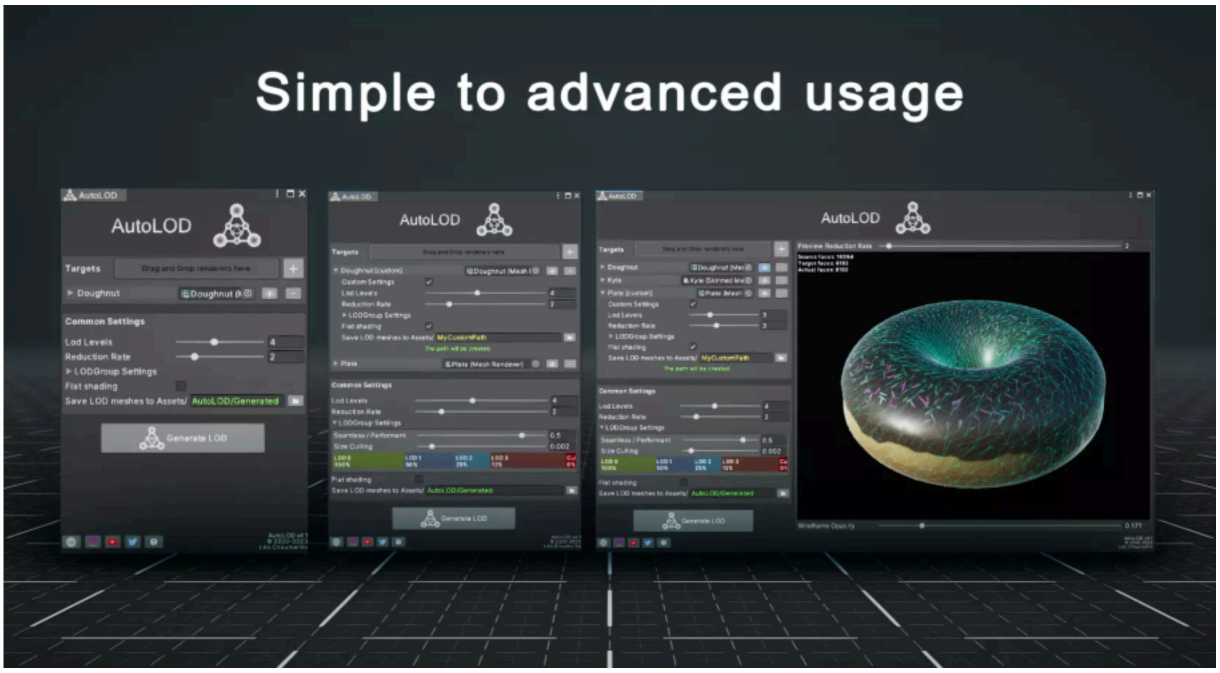
AutoLOD

AutoLOD is a very simple tool that will generate a LOD group with any 3D

object from your scene. It will automatically compute simplified version of your



The AutoLOD editor window under several states of customization

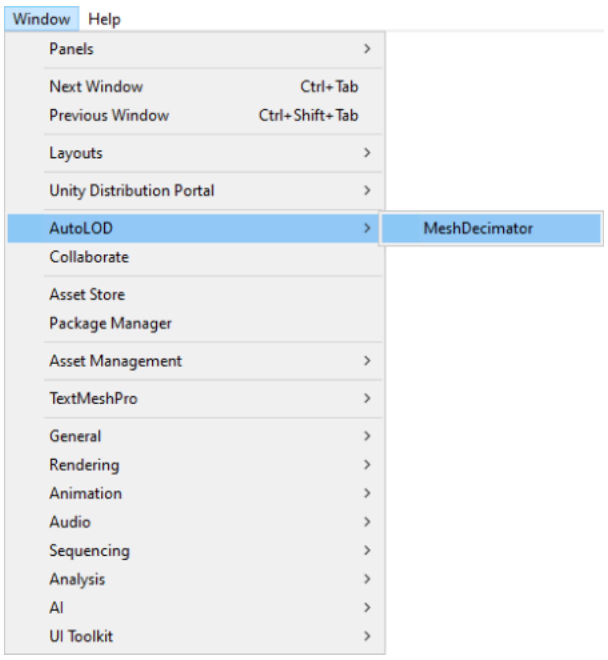
meshes using decimation.

Setup

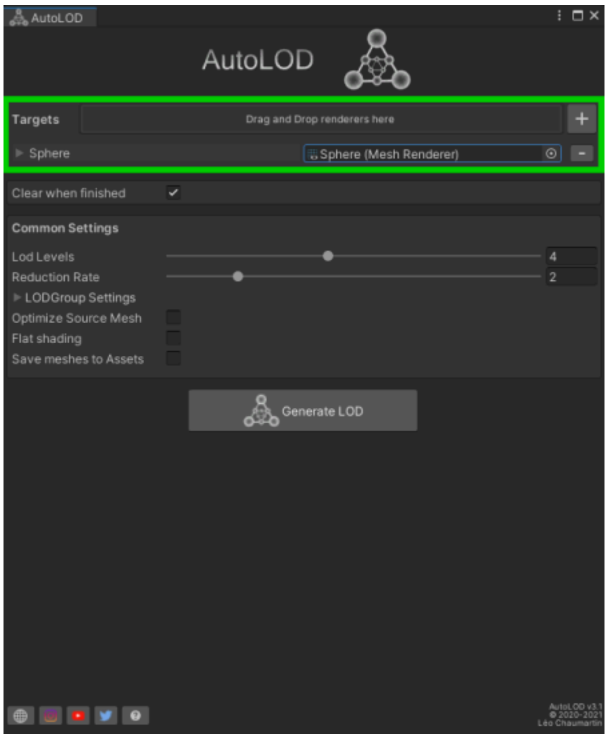
The whole package is contained in the AutoLOD folder resulting from the import. You can move it wherever you want, but it is recommended to keep the same folder structure.

How to use

AutoLOD consists in an editor window. You can open it via `Window->AutoLOD->MeshDecimator`.



Put the object(s) to be processed into the "targets" area. Please note that a Renderer is required otherwise the object cannot be added.

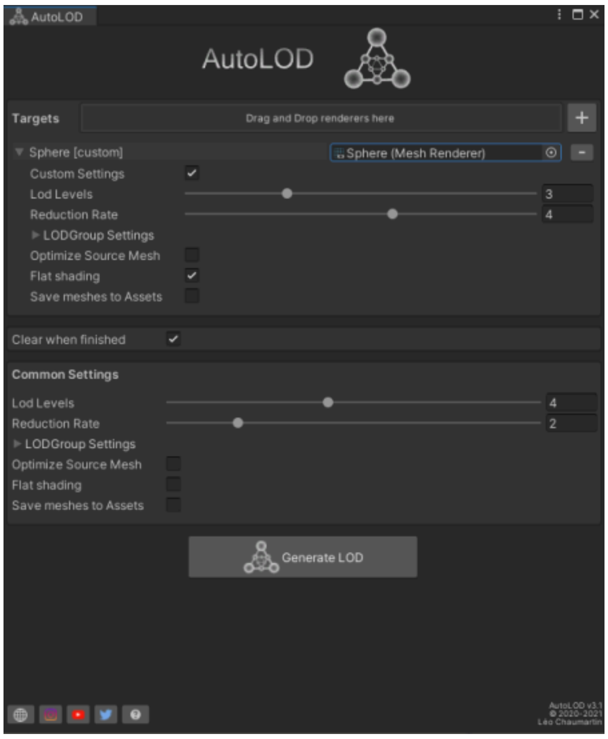


Sphere: Sphere (Mesh Renderer)
Clear when finished: ✓
Common Settings
Lod Levels: 4
Reduction Rate: 2
LODGroup Settings
Optimize Source Mesh
Flat shading
Save meshes to Assets

By default, the same presets will be applied to every renderer in the list.

You can customize these presets in the Custom Settings section.

You can also customize per-object settings by folding out targets in the list.


AutoLOD

Targets — Drag and Drop renderers here

Sphere [custom] — Sphere (Mesh Renderer)

- Custom Settings ✓
- Lod Levels 3
- Reduction Rate 4
- LODGroup Settings
- Optimize Source Mesh
- Flat shading ✓
- Save meshes to Assets

Clear when finished ✓

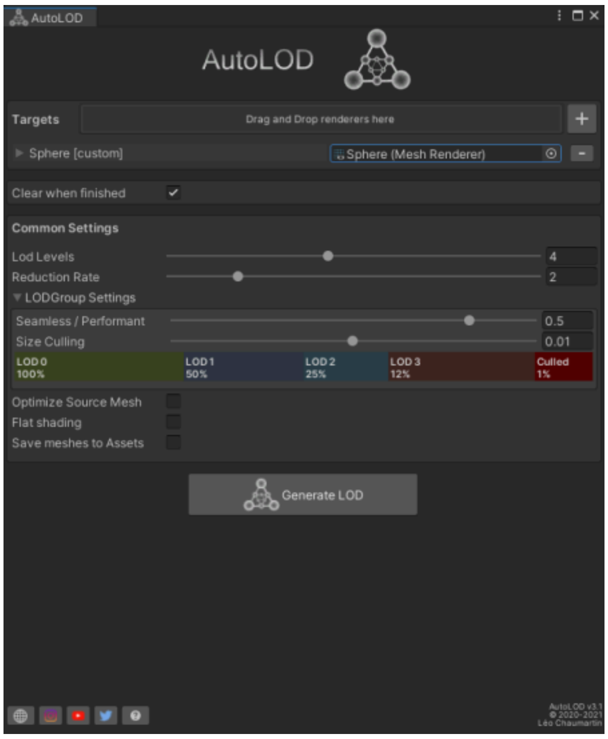
Common Settings

- Lod Levels 4
- Reduction Rate 2
- LODGroup Settings
- Optimize Source Mesh
- Flat shading
- Save meshes to Assets

Generate LOD

AutoLOD v3.1 © 2020-2021 Léo Chaumartin

Both settings panels are identical, meaning that each object is fully customizable independantly from others or from the common settings.


AutoLOD

Targets — Drag and Drop renderers here

Sphere [custom] — Sphere (Mesh Renderer)

Clear when finished ✓

Common Settings

- Lod Levels 4
- Reduction Rate 2
- LODGroup Settings
  - Seamless / Performant 0.5
  - Size Culling 0.01

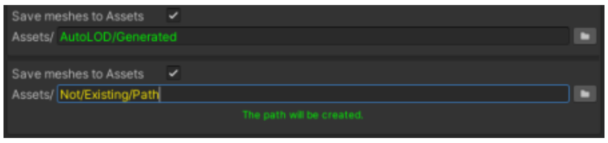
LOD 0	LOD 1	LOD 2	LOD 3	Culled
100%	50%	25%	12%	1%

- Optimize Source Mesh
- Flat shading
- Save meshes to Assets

Generate LOD

AutoLOD v3.1 © 2020-2021 Léo Chaumartin

The LODGroup Settings foldout let you control the LODGroup setup as it should be when created, but you can still change it manually when the process is over. You cannot interact with the LODGroup widget (which is NOT the actual widget, but just a representation), only the sliders LOD Levels, Seamless/Performant and Size Culling will influence it.


Save meshes to Assets ✓
Assets/ AutoLOD/Generated

Save meshes to Assets ✓
Assets/ Not/Existing/Path
The path will be created.

Since 4.0, decimated meshes will be saved to the assets, to the given path. If the path does not exist, the program will automatically create it for you.

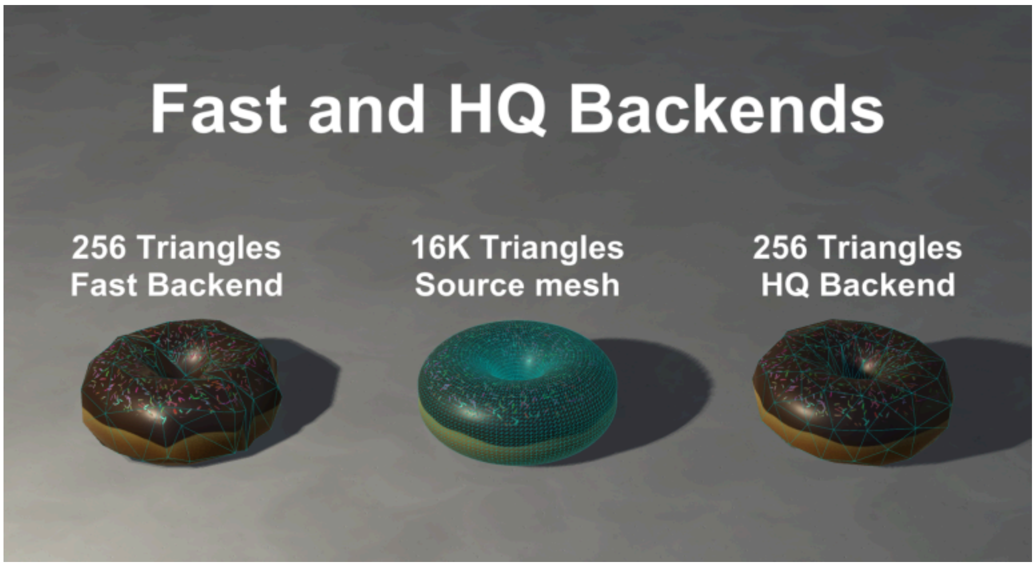
When you are ready, just click on the button `Generate LOD`

Features

- Minimalist interface and really easy to use
- Exposes 2 backends to the users : a fast to be used for prototyping of dynamic usage, and a quality for higher quality outputs (since version 5.0)
- Works in both edit and play mode
- Works on prefabs instances (no need to unpack them)
- You can change the default export path from `Edit->Preferences->AutoLOD Settings`
- Generates up to 8 LOD for a single object
- Decimates UVs (from uv1 to uv4) and bone weights too.
- Compatible with sub-meshes
- Compatible with SkinnedMeshRenderers with bones and blendshapes support
- Can be used for simple distance culling too
- Fully compatible with any render pipeline and any unity version since 2020.3 LTS.

Backends

Since AutoLOD 5.0, you can now choose which backend you want to use between the fast or high quality. Both backends implement Fast quadric mesh simplification but the high quality preserves better UVs and preserve borders.



AutoLOD embedded fast and HQ decimation backends

Runtime API

All the following methods are part of the `AutoLOD.MeshDecimator` namespace.

`void CFastMeshDecimator.Initialize()`

Initializes the MeshDecimator with the Fast backend. This method sets up necessary components or variables required before the decimation process begins.

`Mesh CFastMeshDecimator.DecimateMesh(Mesh sourceMesh, int targetFaceCount, bool interpolateBlendshapes = false)`

Decimates the provided mesh (*sourceMesh*) to a specified number of faces (*targetFaceCount*) using the Fast backend.

`static Mesh CFastMeshDecimator.DecimateMeshStatic(Mesh sourceMesh, int targetFaceCount, bool interpolateBlendshapes = false)`

Provides a static method to decimate a mesh without needing an instance of the FastMeshDecimator class. It takes a mesh (*sourceMesh*) and decimates it to the specified number of faces (*targetFaceCount*) using the Fast backend.

`Task<Mesh> CFastMeshDecimator.DecimateMeshAsync(Mesh sourceMesh, int targetFaceCount, bool interpolateBlendshapes = false)`

Provides a static method to decimate a mesh asynchroneously using the Fast backend.

`void CQualityMeshDecimator.Initialize()`

Initializes the MeshDecimator with the High Quality backend. This method sets up necessary components or variables required before the decimation process begins.

`Mesh CQualityMeshDecimator.DecimateMesh(Mesh sourceMesh, int targetFaceCount, bool interpolateBlendshapes = false)`

Decimates the provided mesh (*sourceMesh*) to a specified number of faces (*targetFaceCount*) using the Quality backend.

`static Mesh CQualityMeshDecimator.DecimateMeshStatic(Mesh sourceMesh, int targetFaceCount, bool interpolateBlendshapes = false)`

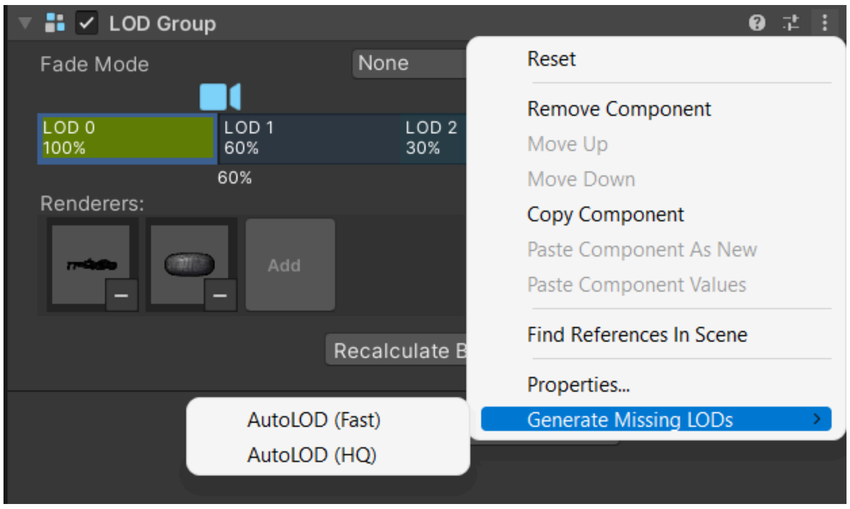
Provides a static method to decimate a mesh without needing an instance of the QualityMeshDecimator class. It takes a mesh (*sourceMesh*) and decimates it to the specified number of faces (*targetFaceCount*) using the Quality backend.

`Task<Mesh> CQualityMeshDecimator.DecimateMeshAsync(Mesh sourceMesh, int targetFaceCount, bool interpolateBlendshapes = false)`

Provides a static method to decimate a mesh asynchroneously using the Quality backend.

Alternative Workflow

There are scenarios where you may want **a single LODGroup for several renderers**, for example a props cluster. AutoLOD 5.2 brought a LODGroup-oriented workflow for this usecase. Just add a LODGroup (preferably on a common parent to all renderers you want to include) and add all renderers you want to include in the cluster in the LOD0 level (required). From the contextual menu, you will be able to generate all the missing LOD levels.



The reduction rate will be proportional to the transition height you entered in each LOD level, following this formula:

$$ReductionRate = 1/TransitionHeightRatio$$

Both backends are available and a progress bar will keep you informled during the process.

If you want to fine-tune some LODs specifically, just delete the LOD's renderers and regenerate from the contextual menu. The LODs that already have renderers will be ketp untouched.

Please note that the meshes decimated using this method are saved in the default export folder that you can customize in the preferences `Edit->Preferences->AutoLOD Settings`

Some tips

- A demo scene is provided in the Scene folder: click play and follow the instructions. The provided material in the scene should appear pink in LWRP/URP and HDRP. Feel free to upgrade the material to your render pipeline, or use your own material.
- Setting 1 on LOD Levels will just apply a distance culling (and an automatic decimation if the toggle is enabled).
- You can apply changes to a prefab instance, as long as the decimated mesh were saved to the Assets (otherwise the meshes will be missing, as they are stored in the scene data).
- You can undo the whole AutoLOD process with Edit -> Undo
- You can put several objects in the drag and drop area. AutoLOD will just add the valid ones to the list.
- You can add a MeshRenderer or SkinnedMeshRenderer from its context menu in the inspector

Warnings

- Generation can take a long time for a whole scene. Unity will freeze during the process but a progress bar will provide some progress report. The progress bar is more precise for Unity 2020 and above than the previous releases.

- If the processed object has specific components attached to it, they will remain but make sure they still work properly. If they were using the attached MeshFilter or Renderer, they will probably be broken, but it should be easy to fix in most cases.
- Please check carefully which objects you put in AutoLOD and understand that there are just too many cases that can end up with bugs, especially with your MonoBehaviour scripts.

Contact

Feel free to contact me via one of the social media available on the bottom of

the AutoLOD editor window or via e-mail: support@leochaumartin.com (mailto:support@leochaumartin.com)

References

https://github.com/Whinarn/MeshDecimator

https://github.com/crescent3983/UnityMeshDecimation