Voxeliser Documentation

Voxeliser

The Voxeliser is a shader-esque script that converts any model mesh to be made up of voxels(cubes) as seen adjacent.
Depending on settings it can affect static or animated models in real-time.

Initial Setup

The voxeliser is split up into two main scripts.
Voxeliser.cs and Voxeliser_Burst.cs, both perform the same action however Voxeliser_Burst.cs utilises the Burst, a recent addition to Unity to optimise how the Voxeliser runs. In almost all cases it will perform 3-5 better. To use Voxeliser_Burst.cs you will need the dependencies as listed below.

Dependencies

Voxeliser.cs

No needed Dependencies.

Voxelsier_Burst.cs

Unity.Mathematics V1.1.0 or greater.
Unity.Burst V1.1.2 or greater
Unity.Collections V0.1.0 or greater

Editor/Runtime Setup & Variables

Only a single script should be assigned per game object. Both scripts have the exact same variables.

Voxel Size:

How large each voxel should be in units.

Voxeliser Type:

What type of voxelising is intended, as described below:

Solid:

Works on any mesh that uses a mesh filter. Will snap voxels onto a world grid regardless of transformation.

Dynamic Solid:

Works on any mesh that uses a mesh filter. Designed for a mesh filter that will undergo modifications, such as movement of vertices via animation or code.

Animated:

Works on models that use a skinned mesh renderer, typically for animated models. Will snap voxels onto a world grid regardless of transformation.

Static:

Creates a static mesh intended to remain in a static position or simply as a conversion. At initialisation will make the correct model mesh, but after, any transformations won't cause voxels to snap to the grid. However this leads to little or no overhead.

Perform Over Frames:

In assisting with optimisation, calculations can occur over several frames as needed. This can lead to smoother gameplay but add more stutter to any animation.

Object With Mesh:

This is the object that contains the model mesh. If not assigned it is assumed to be on the same object as the script.

Delayed Initialisation:

For any particular reason a delay of a single frame is needed before initialisation. Typically used when other meshes are generated at runtime and order of execution is unknown.

Total Mesh Count:

Due to the limitation of max vertices per mesh, several meshes can be used at once to get extra large or complex models.

Vert Type:

Soft Edge:

Voxels are made like a normal cube, 8 verts that are shared on each side. This can lead to some minor lighting artifacts due to the corner normal not being aligned with the face normal, but rather the average of the three connected faces.

Hard Edge:

This increases the count of verts from 8 to 24, with each face having its own unique 4 verts. This removes the lighting artifacts as each verts normal now follows the face's normal.

Save Static Mesh:

When building a static mesh, the mesh can be saved for later use rather than using the voxeliser every time. This variable when toggled in the inspector will create a save prompt, save the mesh. This only occurs when using the unity editor.

Reset Save Rotation

When saving a static mesh, should the rotation be ignored?

Additional Information/ Known Issues

In general Voxeliser Burst will run better.

However when using Burst, jobs inherently cannot run for longer than 4 frames, given Voxeliser Burst includes two jobs to complete, when performing over multiple frames it will still be limited to a max of 8 frames, whereas Voxeliser can perform indefinitely.

In the case of multiple skinned meshes each should have its own voxeliser attached. Issues can occur where meshes overlap, as voxels will also overlap, creating faces in the same location.

Due to a limitation in unity meshes, there can be no more than 65535 vertices in a single mesh. As each voxel is 8/24 vertices, this leads to a hard limit of 8191/2730 voxels per object

depending on the vert mode. In the case of needing more, the total mesh count can be increased.

It currently does not work when using multiple submeshes, it can convert the mesh but will use only the first material.

When using Unity 2017.4 version, you are unable to save a static mesh. This feature will be added at a later date.

When using Unity 2017.4 version, errors will occur when first installing. This is due to the assembly file looking for packages it can't find due to being out of date. Simply deleting the assembly file, or disabling the GUID checkbox will fix this issue.