Voxel Destruction Package

Install Unity Jobs and Unity Collections Package

This Package is using BetterStreamingAssets and VoxReader, licensed under MIT

It doesn't work on Android without BetterStreamingAssets, but it is possible to remove it (Visit VoxelObject.cs)

Setup

Note: It is recommended to turn off Vsync

If you want to create a Voxel Object:

- Model it in MagicaVoxel and save the .vox file in the streaming Asset Folder (make sure it only has small letters in the name)

- Create a new GameObject, Add the Voxel Object Component, enter the Path, set the Material to the VoxelModel Material, and select Build Model, now there should appear the model in unity

- If the Model appears in a violet Color, the Material Shader is not compatible, visit the section bellow about Render Pipelines

- Now you can change all the values to perfection

Note:

- The Editor Preview of the Model doesn't have a collider

- Performance is much better on Builds

If you want to make an Object collidable with Voxel Objects:

- The Objects needs to have a Rigid body with a collider

- Add the "Voxel Collider" script to the Object with the Rigid body

- You can increase or decrease collision scale for more or less destruction

How to improve performance:

- Turn off Vsync

- Split your object into smaller models in MagicaVoxel (every piece then gets drawn separately which makes everything faster)

Render Pipeline

For the Voxel Models you just need a Material Shader that supports Vertex Colors, like the Particle Shader.

By default, the Package is designed for URP (Universal Render Pipeline)

If you are not using URP, you will get some warnings about missing components, this is caused by some URP scripts, and you can just remove them

These missing components are found on the following Objects and can be removed if you don't want to use URP:

- Directional Light

- Main Camera and Overlay Camera (Childs of the Camera Object)

Built-in render (legacy) Pipeline:

Go to "/Voxel Destruction/Material/Built-In" and assign the Ground Material to ground (in demo scene)

Select every Voxel Object and set the Material Property to the legacy Voxel Material in that Folder

HDRP (High-definition Render Pipeline):

If you're using HDRP you need to create a new Shader Graph, create a Vertex Color Node and connect it to color, you can adjust the smoothness to your needs

Create a new Material based on that Shader and set the Material Property on every Voxel Object to the new made Material

Parameter description

Parameters of Voxel Object script:

Path: The vox file path relative to the Projects streaming assets

Build Object: Defines if the Object should be created on awake

Export Mesh: If set to true on playing an editor window will allow you to save the model as an Asset

Start With Physics: If set to true a convex collider gets created and the Model will immediately have a rigid body added to it.

Create Collider: If set to true a Mesh Collider gets added to the Object

Use Convex Collider: If set to true the Mesh Collider will be set to convex

Add Voxel Collider: If set to true a Voxel Collider gets added which allows for Voxel intercollision

Collision Scale: The collision scale if of the Voxel Collider, this option will only show if add Voxel Collider to true.

Build Entire: If set to false you can define which model should be built, only useful if multiple models are in the Vox file.

Object Scale: The scale of a single Voxel in the Mesh

Loading Time: The maximum Mesh calculation time in frames

Delay Recalculation: This option delays the Mesh recalculation and is recommended if you are using Vsync. However, performance is much better if you turn off Vsync and this option.

Use 32 Bit Int: If set to true a 32-bit int is used for Mesh calculation, this allows for bigger Meshes but reduces performance

Generation Type:

- Safe: Simple IJob is used for Mesh calculation, slow but consistent
- Normal: IJobFor is used for Mesh calculation, with only 3 calls
- Fast: IJobFor is used for Mesh calculation, large number of calls

Schedule Parallel: Runs the Mesh Job parallel on multiple Threads, not recommended as it can cause calculation errors and is not very persistent

Allocator: The Allocator used for the Mesh Job, use either Persistent or TempJob. Persistent is recommended as it is more stable, but TempJob is faster but can cause errors if the calculation is longer than 4 frames.

Total Mass: The total mass of the Voxel Object, is used to set the Rigidbody mass of the Voxel Object and to calculate the Fragment mass.

Drag: The Rigidbody drag of the Voxel Object and Fragments

Angular Drag: The Rigidbody angular drag of the Voxel Object and Fragments

Constraints: The Rigidbody constraints of Voxel Object and Fragments

Interpolation: The Rigidbody interpolation of Voxel Object and Fragments

Collision Mode: The Rigidbody Collision Mode of Voxel Object and Fragments

Destructible: Defines if the Object should be destructible

Destruction Settings:

Destruction type: Simple Enum with the following options:

- Passive Voxel Removal: The Voxel gets removed from the Mesh, but no fragments or separate Voxels are created.
- Active Voxel Removal: The Voxel gets removed from the Mesh, and a new cube a is created for replacing the Voxel. Note that this one is only useful with small Models at it is very performance heavy.
- Fragmentation Removal: The destroyed Voxels get removed from the Mesh and are replaced by a Fragment.

- Precalculated Fragments: The fragments get calculated and created at Start and then activated once needed. This is very efficient as it requires not much calculation on playtime but can take some time loading at the beginning.

Min Collision Magnitude: The minimum collision Magnitude to trigger destruction

Collision Strength: The Voxel removal count calculated from the collision magnitude

Collision Force: The Force applied to the Fragment Rigidbody on destruction

Physics On Collision: Makes the model a physic object on collision

Destruction Sound Settings:

Collision Clip: A string array containing the names of sound effects that play on collision, a random one gets selected. Make sure that the sounds are setup in the Sound Manager, if you don't want to use any sounds just clear the array.

Sound Volume Scale: The Volume Scale calculated the collision magnitude

Destruction Voxel Removal Settings:

Max Voxel Distance: The maximum distance between the collision point and the destroyed Voxel

Particle: The particle spawned on destruction, needs to have the Particle system as the first child

Destruction Fragment Settings:

Relative Size: If set to true the Fragment size is calculated from the collision strength, this option is only necessary when using Fragmentation Removal

Relative Fragment Scale: The scale used to calculate the fragment size from the collision strength, this option is only necessary when using Fragmentation Removal

Minimum Fragment Size: The minimum fragment Voxel count if you are not using relative Sizing, this option is only necessary when using Fragmentation Removal

Maximum Fragment Size: The maximum fragment Voxel count if you are not using relative Sizing, this option is only necessary when using Fragmentation Removal

Destruction precalculated Fragment Settings:

Use Job Fragmentation: Defines if a Job should be used to calculate the fragments, is recommended for Object with many fragments as it increases performance.

Fragment Sphere Radius Min: The minimum radius of the sphere used to calculate the fragment, a random value between min and max radius gets selected

Fragment Sphere Radius Max: The maximum radius of the sphere used to calculate the fragment, a random value between min and max radius gets selected

Errors

If you keep getting errors, try the following things:

- Make sure the Collection and Burst package is installed (Package Manager)

- If you're getting Errors about a shader and are not using URP, delete the "/Voxel Destruction/Material/URP" Folder

- If you're getting Allocation Errors on Runtime, switch the Allocator of VoxelObjects to Persistent

- If your getting a delay between the creation of Fragments and the recalculations of the Mesh you can try to turn of Vsync, set the maximum loading time to 1 or disable "delay recalculation".

Support email : atangameshelp@gmail.com

How it works

For anyone interested in how this package works, here are some explanations. This is useful for people who want to modify the package or use parts of it for other purposes (for example the MeshBuilder script could also be used to generate Voxel Terrain).

The VoxelObject script starts to draw the VoxelObject in the Start method, first it checks if the .vox file exists and if it does it starts reading it with BetterStreamingAssets on line 146 (If you want to disable BetterStreamingAssets, you can change the second argument on the read method from true to false). The read bytes then get converted to a data structure called "IVoxFile". If you want to learn more about this process, check out VoxReader: https://github.com/sandrofigo/VoxReader

Next the "DrawVoxelFile" Method is called, since .vox files can contain multiple Models all of them need to get drawn separately. There is also the option to only build one model with a specific index.

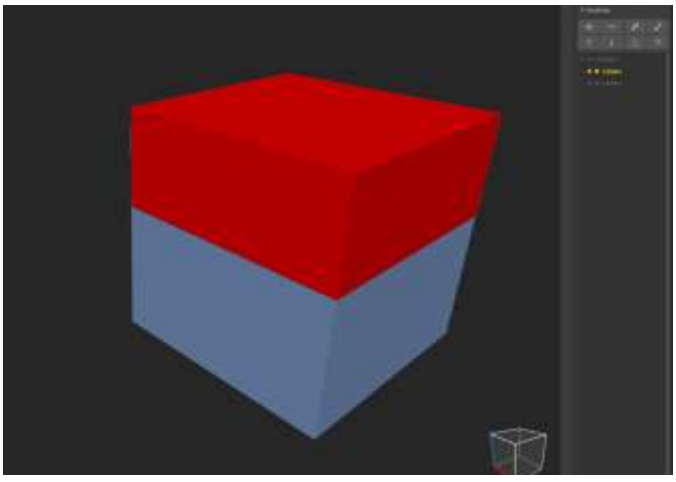

Figure 1.1

The cube in Figure 1.1 is split into 2 parts, the upper part would get the index 0 and the lower part the index 1. In Unity these parts would get drawn separately, this can have benefits, like saved performance since only half of the mesh needs to recalculate, but a disadvantage would be a larger triangle count as greedy meshing is sperate for both models and draws unnecessary faces between them. In the "DrawVoxelFile" method a for loop assigns each Model his own VoxelObject.cs file. The Model with index 0 gets drawn on the script running the for loop, the other ones get drawn on instantiated objects. The Meshfilter containing the Voxel mesh is always a child of the VoxelObject.cs script.

Now the specified model gets drawn, for this the IModel data structure gets converted to the VoxelData data structure. They both contain an array of voxels, the difference being that voxels stored in IModel have a Vector3 variable containing their position, while the position of voxels in Voxeldata can be calculated by their index. This is done to save performance as otherwise it would be difficult to find a voxel on a certain position without looping through the whole array. The disadvantage is that more memory is used because also empty voxels are in this array, each voxel has a bool to check if it is active.

After converting the vox file to Voxeldata the “CreateVoxelMesh” coroutine gets started, this coroutine is responsible for calculating the Voxel Mesh and is also called whenever changes to the Mesh were made. This Method uses MeshBuilder, MeshBuilderSafe or MeshBuilderParallel to create the Mesh, they all use a different job to create the Mesh. MeshBuilderSafe uses a IJob, MeshBuilder uses a IForJob with a length of 3 and MeshBuilderParallel uses a IForJob with a length relative to the Models size. There also is to option to schedule MeshBuilderParallel or MeshBuilder parallel, but it causes some weird index errors, because of multiple cores accessing the same array the same time. All have a “StartMeshDrawing” method to start the drawing and a “GetVoxelObject” method to get the Mesh. Make sure to call the “Dispose” method afterwards.