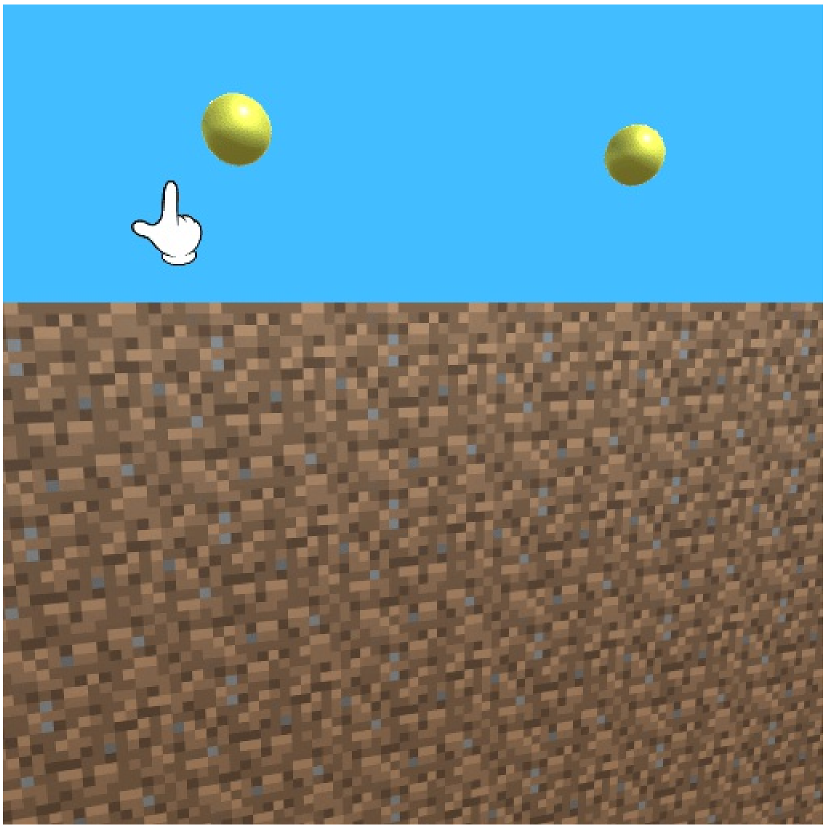
Unity Pixelated Digging

A pixelated digging system with textures and runtime mesh generation in Unity.

This system utilizes the basic idea behind **marching squares** algorithm with slight modifications.

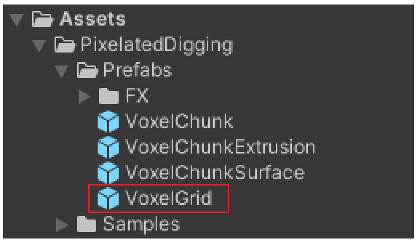


You can find the original source code here:

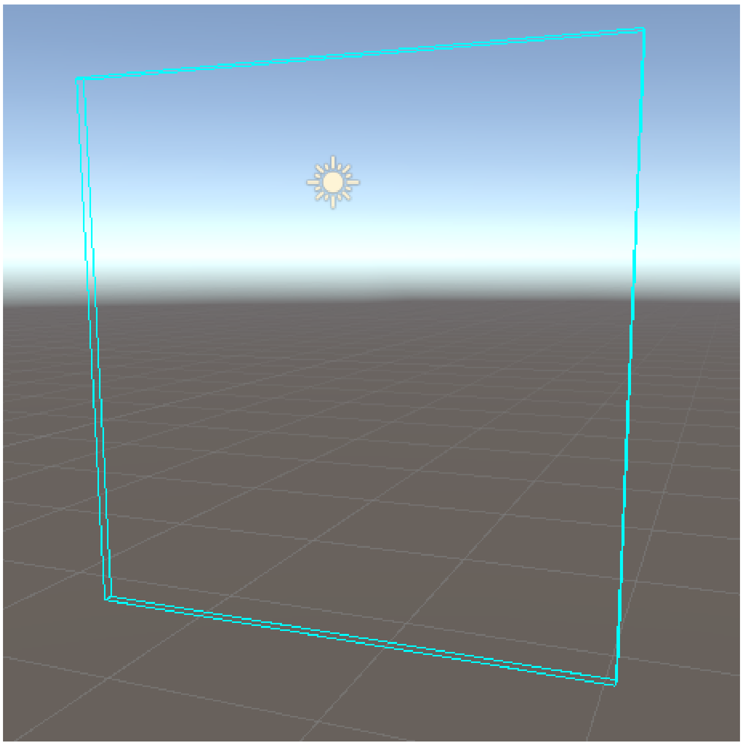
- https://github.com/demircialiihsan/unity-pixelated-digging

How to Use

After adding the package into your Unity project, find the **VoxelGrid prefab** in the path **Assets->PixelatedDigging->Prefabs->VoxelGrid** and drag it into your scene.



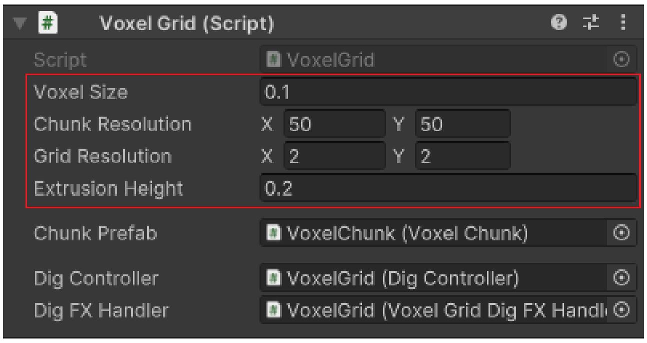
You may now see the gizmos of the **VoxelGrid object** in the scene view. It shows how the grid will look like once it gets initialized in play mode.



Chunks

Once initialized, *Voxel Grid* consists of **Chunks**. This way, whenever digging occurs, only the relevant *chunks* update their meshes.
Each *chunk* comes with 2 meshes, one for the surface and one for the extrusion parts.

Select the *Voxel Grid* object from the hierarchy to reveal the **VoxelGrid** component in the inspector. Here you can see some fields of the *VoxelGrid* that you can change as you would like. Notice how the gizmos change as you tweak these values.



Voxel Grid (Script)

Script VoxelGrid

Voxel Size 0.1

Chunk Resolution X 50 Y 50

Grid Resolution X 2 Y 2

Extrusion Height 0.2

Chunk Prefab VoxelChunk (Voxel Chunk)

Dig Controller VoxelGrid (Dig Controller)

Dig FX Handler VoxelGrid (Voxel Grid Dig FX Handl

- **'Voxel Size'** defines the size of each **voxel(pixel)** on both X and Y axis in units.
- **'Extrusion Height'** defines the size of each *voxel* on the Z axis in units.
- **'Grid Resolution'** defines the number of *chunks* on the grid for each axis.
- **'Chunk Resolution'** defines the number of *voxels* on each *chunk* for each axis.

In this example the grid will consist of 4 *chunks*(2x2), and will have 100 *chunks* on both X and Y axis(2x50). Each side of the grid will be 10 units long(100x0.1).

About the Algorithm

As previously mentioned, this digging system is inpired by *marching squares* algorithm but has some fundamental differences.

In traditional *marching squares*, control nodes are placed in the corners of cells, whereas in this system, control nodes are the *voxels* themselves and they define a cell at the center of which they are located.

References

- https://catlikecoding.com/unity/tutorials/marching-squares-series
- https://github.com/SebLague/Procedural-Cave-Generation