

Assignment 02 – Minecraft – Due 10/01/2019 @ 9:00 AM

Submission Format:

All projects must be zipped (.zip) into a single file. Submit your assignment via drop box / google drive to your instructor with the following naming convention: ##_\$\$ where ## is the assignment number and \$\$ is your full name.

Example: 02_GLecakes.zip

Both the zip file and the subject of your e-mail should adhere to this convention.

Failure to follow this submission format will result in a zero for your grade.

If a submission is late, you will receive NO CREDIT for the assignment.

Assignment Goals:

This assignment is meant to get you acquainted with the model creation through code. You will explore how to manipulate vertices, UVs, normals and indices to create simplistic models procedurally.

Grading:

This assignment counts as a portion of your homework grade. You will be graded based on your ability to make this assignment work. Further down is a list of expectations with bullet points. I will use this list to determine your final grade, with each bullet point being worth 1 point.

Assignment Requirements:

- Create an empty game object and attach a new script called Chunk and do the following
- Require a mesh filter and mesh renderer be on the Game Object
(<https://docs.unity3d.com/ScriptReference/RequireComponent.html>)
- Store the mesh renderer / mesh filter as variables inside the class
- Check to see that the mesh renderer and filter exist and if they do not, attach them through scripting in your start method.
- Find and download a Minecraft atlas texture for the blocks using a search engine. It should look like Figure 1 at the end of this document. Add it to your Unity project under a folder called Textures.
- Create a three-dimensional array and populate it with random values of 0 through 4 (inclusive)
 - Air = 0
 - Dirt = 1
 - Brick = 2
 - Wood = 3
 - Ore (your choice) = 4
 - (<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/arrays/multidimensional-arrays>)

- <https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netframework-4.8>
- Create a method that iterates over the 3D array and checks each value
- Create a method called 'CreateBlock' that will create the information for making a cube. Pass in the offset in the 3D array the block is at and the type of block it is. The offset should alter the vertices to be in the correct location. The block type should use the correct UVs to map the right part of the texture to the polygons.
- You should 'hard-code' the vertices and UVs for a block and use vector addition and the offset into the array to slide the vertices and UVs to the correct location.
- Determine where each of the previously mentioned blocks are on your Minecraft texture and assign UV coordinates to each (store these away for the 4 corners of your quad for use in assigning UV coordinates to your mesh)
- If the value is 0, skip it, otherwise make sure you have the correct texture on the surface.
- The cube should generate vertices, uvs, normals and indices, which need to be combined into one final mesh
- Make sure you attach the texture to the mesh through a material.
- When play is pressed, a series of cubes should be created that matches the random 3D array of data
 - I recommend hard coding the 3D array and then moving onto randomization after you prove generation of cubes works properly.

Graduate Student Class Requirements:

Graduate students will complete all undergraduate portions of the assignment in addition to the following:

- In the previous case, redundant polygons are created. When two cubes of the same type are drawn next to one another there is no reason to draw a dividing polygon between them. This slows down performance and creates unnecessary polygons. Instead of creating just a cube at each location, check to see if the cubes surrounding that location are empty or not. If they are the same, you should not draw that face. Consider breaking the cube method into methods for each side of the cube and checking the surroundings before drawing the face.

