Project 1: Terrain Creation
CS 4803 PG, Spring 2020

In this project you will create a synthetic terrain that includes random height variations. In particular, you will use Perlin noise to create terrain that has both large and small height variations. You will add color to your terrain using a texture map. You will also create new terrain on demand, when the camera moves too far. Finally, you will add some very simple plants to your terrain. Your terrain can be inspired by any number of possibilities, including: desert landscapes, old mountains like the Appalachians, new mountains such as the Rockies, underwater seascapes, alien planets, or island chains.

Project Description

You will create a Unity project that creates a height map terrain. That is, you will create a fine square grid that mostly extends in the X and Z direction, but that also has variations in the vertical (Y) position of its vertices. (Of course each square in the grid is actually two triangles.) Make sure that you use a high resolution grid of squares to create your terrain. If you use a low resolution grid, your terrain will not look good. Unity allows 16,384 vertices per mesh, so you should be able to create a grid that contains 85 x 85 squares. Unity can easily draw a large number of such complex meshes, so you may well decide to put multiple of these terrain grids next to each other at any given time. (Note that this issue is related to creating terrain on demand, which is discussed below.)

Your terrain will have high and low places (e.g. mountains and valleys). You will create these elevation variations by moving the vertices of your grid in the Y direction. You should use the built-in Perlin noise function to create these height variations. If you were to use just one band of Perlin noise, your terrain would not look particularly realistic. You should use at least **three** different bands of Perlin noise, of different spatial frequency and different vertical displacement (amplitude) to create the variations in elevation. This will give your terrain fractal-like qualities, just as is found in natural terrain. Note that the Unity 2D Perlin noise has noticeable repetition in it near (0, 0). To avoid this, add large offsets to the X and Z values that you pass into the noise function.

In order to add color variation to your terrain, you will make use of one or more texture maps. By coloring the pixels of the texture map differently, this will cause various parts of your terrain to have different colors. When you create the triangle mesh for the terrain, you will need to assign **u** and **v** coordinates to your mesh. This will tell Unity which part of the texture map should appear on which part of the terrain mesh. You can choose to color your terrain based on any scheme that you think is appropriate to your theme. For instance, you may want to take the height of the mesh into account for selecting colors. You may want to add small amounts of noise to your colors, to add variety. You may also decide to use different colors in different regions.

Simply creating terrain using a fixed square grid will not give you a very large world in which to move. For this reason, a major component of this project is for you to create additional terrain on demand. This means that when the camera moves close to the edge of your terrain, you should create additional terrain that is a natural extension of the current terrain. The provided example code shows how to add geometry when the camera moves in the positive Z direction. Your code will add new terrain not only in the Z direction, but in which ever direction the camera has moved. The easiest way to create terrain on demand is to create your new terrain pieces in large square chunks.

When you set two terrain pieces side-by-side, it is easy to get their elevations at the vertices along the shared seam to be the same. You can do this by using the world-space X and Z coordinates as the parameters to the noise function. It is more difficult to get the surface normal at these shared seam vertices to be the same. One way to do this is to have your program calculate the surface normals at these seam vertices explicitly.

In order to give your terrain more variation, you will place very simple plants (trees or bushes) at various random places across the terrain. Because this project is not about plant creation (that project comes later) you must only create very, very simple plants that are made up of either **one** or **two** Unity primitives, without texture. Please do not use a custom mesh for your plants. You may decide to create more plants at certain locations based on the terrain characteristics, such as placing more in the low-lying regions. You might also consider varying the colors and the sizes of your plants, perhaps based on where they are placed.

Here is a list of the required elements for the project:

- Create your terrain using at least **three** bands of Perlin noise.

- The triangle mesh for your terrain should not contain cracks or intersections between triangles.
- Be sure that there are no visible seams where two of your terrain grids meet.
- Add **color variations** to the terrain using texture maps.
- Create more terrain **on demand** according to the camera movement.
- Place **simple plants** across the terrain (one or two primitives per plant).
- Allow camera movement, as in the provided sample code

Possible Additions (Not Required)

Below are some ideas about possible additions to your terrain. These are not required elements, but are ideas that you may consider if you feel inspired to go beyond the project basics.

- Create water features automatically, such as rivers or ponds.
- Make other terrain features such as meteor craters, cliffs, or volcanoes.
- Have your project remove terrain from regions that are far from the camera.
- Create rocks or boulders in certain parts of your terrain. (This is hard to do well.)

Additional Rules

As with each of the projects in this course, all of the objects in your scene should be created by you from within Unity. You should not include game assets that have been made using other programs (Maya, Blender, etc.), nor should you include assets from the Assets Store.