**Procedural modelling tutorial 1: First actual landscape work**

Welcome to all to this first practical session. Firstly some notes about the practical example you are being given here.

The basis of this tutorial is raster-tek DirectX 11 terrain example #3. The reason for this is that this tutorial has enough basic content to show a meaningful generation and experimentation of procedural landscapes with the algorithms we discuss during the lectures. Think of this as a sandbox to allow you to experiment initially before you create your own terrain object.

Third years will have their own application framework from semester 1 which they should be familiar with. It is recommended that you add a terrain type object. Since you already have a basic object in Pauls framework for a flat plane, it’s a simple evolution of this. I do not recommend you base your coursework on this included example after you built your framework up last semester.

{Masters students:} Again I recommend you adapt the terrain or create your own terrain specific functionality for your applications created last year.

What is a Terrain?

In our case a terrain is a flat piece or geometry in the X/Z plane. This can be represented by a 2D array of Vertices. Typically the X and Z coordinates of each vertex on the terrain match up to the position of the vertex in the array.. ie Terrainarray [4][6] x= 4 y=0 z=6 .

When the terrain is initialised the y values are set to zero and when we generate a terrain we are going to alter the Y values to create something more visually appealing.

Note that every time you generate or alter the terrain the normal of the terrain will have to be re-calculated to reflect the updated geometry.

Also note that in the example, we are simply manually altering the vertices before they are sent into the pipeline. Given the power and functionality of modern vertex and pixel shaders (not to mention the Tessellation Pipeline) there are multiple other ways of creating and altering the geometry of the object you create. I leave it up to you to determine the best solution. At this stage we are interested in experimenting with generation techniques.

This example features: user input and rendering of basic geometry with lighting and shading. Although the raster-tek tutorials provide an interesting range of terrain rendering techniques, none of them actually procedurally generate the terrain from scratch so certain modifications were made so that the example code serves as an appropriate platform for our work. The raster-tek demo on the website only loads in a height-map rather than uses an algorithm to generate the terrain.

I have added a method to the terrain class called “GenerateHeightMap()” which is where you will be doing your work today. And a new initialisation method “Initialize Terrain“which creates a flat initial terrain of the desired size and replaces the raster-tek “initialise” method which loads in a height map. I have left the height map loading code in there as this can be useful for debugging shaders etc by forcing particular examples.

If you run the example, it will present you with a dull flat terrain geometry and hitting the space key will generate the terrain using the “generateHeightMap” method. This will create a simple Sin-Wave terrain just to illustrate it’s doing something. There are many aspects of the example that can be improved upon .

Some quick-find notes:

The terrain is initialised in ApplicationClass::Initialize

The input-generation call is in ApplicationClass::HandleInput

The new terrain generation and initialisation methods are in TerrainClass::InitializeTerrain

And TerrainClass::GenerateHeightMap respectively.

Tasks-

1: Modify the existing generation algorithm, try changing the direction of the SIN save from the x axis to the Z axis, or change it to a COS wave. You can also change the amplitude or the period of the wave. Something that will massively increase the appeal of the terrain will be to add in a second wave in the other axis along with the initial wave, this will create a basic lumpy terrain.

Note that if you change the AXIS of the wave you may find your lighting becomes very subtle or not visible at all, this is due to the light shining directly down the X-axis. So change the lighting information also.

2: Random height field. Add in an algorithm to generate a random height field. Its recommended that you create a new method to achieve this (which will be called from inside the Generate method)

3: Terrain Smoothing : Implement a method of smoothing the terrain by holding a key down. While it is held down the smoothing will continue until released.

Note: Combinations of randomness and smoothing can actually achieve interesting results. There is no “right” way to achieve the results.

4: At this stage I would consider implementing a terrain style object in your previous framework before you go any further.

5: Although the random method is probably enough for this lab. If anyone gets this far the next step is to think about a faulting or particle deposition method. The basic techniques are discussed in the lecture material but ask myself or the lab helper for details on how to begin implementing these.