CGRA 350 - Project Report

Jackson Tume – 300446258

Github Repo: https://github.com/Blukzen/CGRA350-Group-Project
Link to video showing off my contribution: https://youtu.be/wqc0pYqosN8

Basic Information

Island with birds and cool sun

Team ~Gunsplosion~

Members:

- Braden Roberts-Letiu 300419977 robertbrad1@myvuw.ac.nz
- Jackson Tume 300446258 tumejack@myvuw.ac.nz
- Eugene Park 300413411 parkeuge@myvuw.ac.nz

Abstract:

We plan to combine procedurally generated terrain, with uniformly soft shadows to create realistic and pleasing shadows. Simulated flocks of birds were also incorporated to bring the island to life. My individual topic was procedurally generated terrain.

Group Summary

Name:	Technical Contributions/Responsibility:
Eugene Park:	Soft Uniform Shadows
Braden Roberts-Letiu:	Wildlife Bird Simulation
Jackson Tume:	*Procedurally generated terrain*

Introduction to your individual topic

Overall objective and problem definition:

I wish to create a realistic looking terrain, with the ability to have mountains, rivers, and everything in between. The terrain should be coloured appropriately as so the mountains should have snow colouring if at a certain height, and below sea level the terrain should be coloured as water. The program should also run in real time, with the ability to adjust the terrain to get it exactly how you wish and see the changes in real time.

Technical problems and tasks:

There are a few technical problems with this, as it's a requirement for the program to run in real time. I cannot do calculations pre-hand before rendering, the calculations will have to be done in real time as the program renders the scene. Another difficulty is just how many calculations must be run before the program can be able to render the scene. Every time the user changes a value, the function that computes the actual values of the height map is run, this means any time the user changes a value, there will be a hiccup as the program tries to calculate all the data it needs.

Summary of background and related works with clear reference(s)

As my original plan was to implement hydraulic erosion algorithms my references talk about it a lot. However I did not get to implement it due to the time constraints, the references do talk about generating terrain using noise functions to get coherent randomness, which is exactly what I did also (just using different types of noise).

References:
https://web.mit.edu/cesium/Public/terrain.pdf
https://www.firespark.de/resources/downloads/implementation%20of%20a%20methode%20for%20hydraulic%20erosion.pdf

Summary of your contributions:

My contributions where to implement a terrain mesh, the height of the y co-ordinate is based off a height map. This height map gets the correct values it should be using based off of certain parameters and feeds them into my implementation of a Perlin-esque noise function. This simulates a terrain with mountains and valleys and can calculate the values quick enough as it looks real time.

Solutions and implementation details of your individual topic

Description of solutions:

As I already know the x and z co-ordinates of the mesh needed, we only need to calculate the y (height). From that y co-ordinate we can calculate the all the vertex information needed (position, normal) and using the CGRA framework, it is trivial to build a mesh when we know all the information. This reduces the number of calculations needed, as we only need to figure out the y co-ordinate of every point every time we make a change. If the heightmap we use has a size of 256x256 this gives us a good blend of having a terrain mesh that's a good size, and also not too large that it takes too long to compute when we make changes. Which would break the illusion of real time changes if it stutters and drops frames every time it updates. Depending on the machine used, a smaller map or larger map can also be used.

Implementation details:

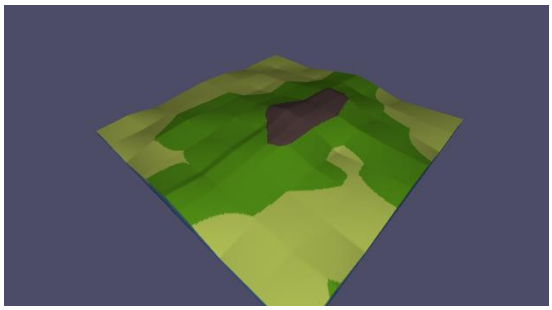
To implement this, I started by having a 2D vector of floats. This vector would be the heightmap, I then check every frame if the user has made an update to the parameters, if they have – then rerun the function with the new parameters to get the new result. Every time the heightmap is updated the mesh is then rebuilt, however we still have the same indices, so we don't need to recalculate them again. Using octaves (number of times the code loops) we can easily get low and high poly looking meshes. The number of vertices is actually the same for all the meshes, regardless of the number of octaves (around ~65k for a heightmap of 256x256). However the appearance will change from looking high and low poly.

Results of your individual topic

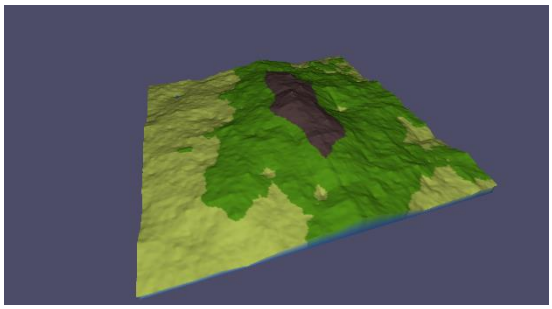
Summary of what you achieved:

I successfully implemented real time procedural terrain generation. That can change in real time according to the user input. The colours of each vertex are set appropriately in the vertex shader as to give the terrain a more realistic and pleasing look. I also added a water mesh which is just a plane that sits at sea level, however on the edges of the map it sets the value to the same as the heightmap, giving the terrain a cool "drop-off".

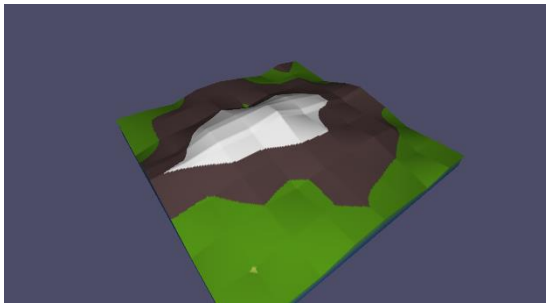
Images of important outputs:



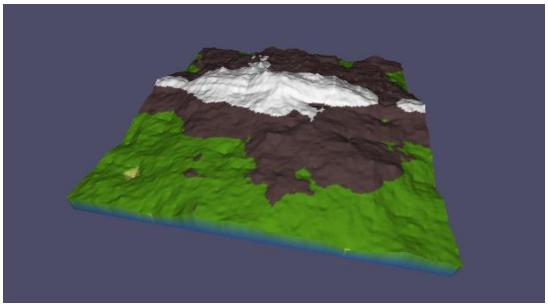
Low Poly look - flat parameters



High Poly look - flat parameters



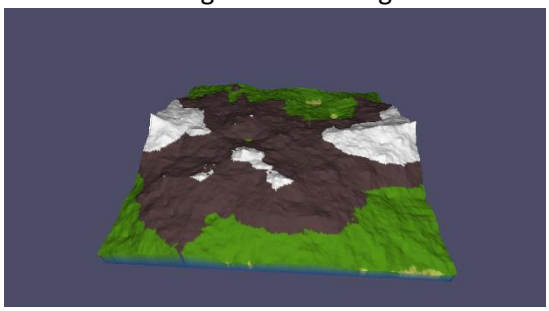
Low Poly look - high parameters

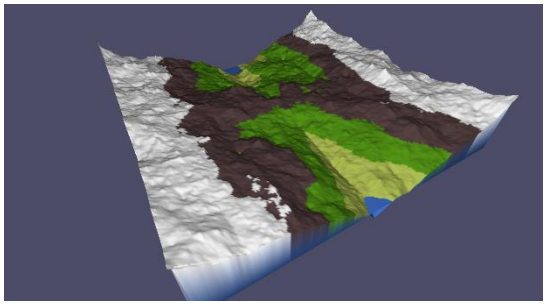


High Poly look - high parameters

The above 4 images all have the same seed, just different parameters. This shows how versatile the algorithm is as it can get completely different results using the same seed, just with different parameters.

The below 2 images are showing off random seeds, to get random but still realistic looking results.





*Attached with the submission are the high resolution versions of each image.

Discussion:

Successes and limitations of what you have done personally:

I successfully personally implemented procedural terrain generation, the limitations of how I have done it, however are. The terrain looks good, but not as realistic as I would have liked. My original plan was to also implement hydraulic erosion which simulations hundreds of thousands of water droplets falling on the terrain, which overtime erode leading to a much more realistic terrain. If I had more time, I would definitely implement this as even though I am very happy with what I have accomplished in the timeframe. I would like to have a more realistic terrain.

Integration with other team members:

Integrating my contributions with the other team members was very easy, as I was using the CGRA framework build the terrain mesh, the entire terrain mesh data could be kept in a single variable. I also kept most of the code I worked on in different folders/files to other members, which allowed me to not worry about breaking any of their code while changing mine. Because of how I implemented everything, it only took a couple of minutes to add the other members code into my project as copying over their files couldn't break anything of mine.