Solent University

SCHOOL OF MEDIA ARTS AND TECHNOLOGY

**BSc Computer Games (Software Development)**

**2019**

**Oliver Mills**

***“Procedural map generation”***

Supervisor : Mark Bennett

Date of submission : 3rd May 2019

Abstract

Contents

[1 Background 1](#_Toc2305278)

[1.1 Overall Aims 1](#_Toc2305279)

[1.2 Objectives 1](#_Toc2305280)

[2 Description of research/prototyping completed 2](#_Toc2305281)

[2.1 Discussion of potential solutions - Algorithms/approaches 2](#_Toc2305282)

[2.1.1 Tectonics 2](#_Toc2305283)

[2.1.2 Erosion 3](#_Toc2305284)

[2.1.3 Hexagonal tile-based generation 3](#_Toc2305285)

[2.1.4 Voronoi Diagrams 4](#_Toc2305286)

[2.2 Discussion of tools and technologies – Criteria 4 25% - 250 words 4](#_Toc2305287)

[3 Project Specification 5](#_Toc2305288)

[4 Discussion of software dev methodology 5](#_Toc2305289)

[5 Discussion of project management tools and metrics 6](#_Toc2305290)

[6 Resource implications 6](#_Toc2305291)

[7 High level overview of classes 7](#_Toc2305292)

[7 High level flow diagrams and pseudocode 8](#_Toc2305293)

[Appendices A](#_Toc2305294)

[Appendix A - References A](#_Toc2305295)

[Appendix B - Reading list B](#_Toc2305296)

[Appendix C - Literature review C](#_Toc2305297)

[Appendix D – reference image F](#_Toc2305298)

[Appendix E - Discussion of tools and tech G](#_Toc2305299)

[Appendix F - Fortunes Algorithm pseudo code H](#_Toc2305300)

[Appendix G - Library Comparison I](#_Toc2305301)

[Appendix H – Risk Analysis J](#_Toc2305302)

[Appendix I - Gantt Charts K](#_Toc2305303)

[Estimated duration assuming only small issues occur K](#_Toc2305304)

[Estimated duration doubled to allow for larger issues K](#_Toc2305305)

[Appendix J - Work Log M](#_Toc2305306)

Progress Report

1 Background

This project will focus on generating maps designed for world settings to simplify and streamline world creation for table top roleplaying games (rpgs), with a focus on fantasy settings similar to ones used in Dungeons and Dragons (D&D) (coast, Wizards of the, 2014). These maps will be procedurally generated one at a time and will have unique terrain and biomes with each different map, with stretch goals of implementing resources linked to the environment and generating settlements based on said resources. Each map will be presented similarly to Skyrim’s world map (fig 1) but using a much lower poly style and simplified representation of terrain types, for example foliage would not be included.

This project is different from other similar projects that already exist due to its focus on being 3D and stylised while others are visually represented in 2D.

This project is being created to gain a deeper understanding of procedural world building techniques. This will be a technically challenging project due to the use of complicated algorithms and processes which must all work together to complete different end products and thus a good fit for the course.

1.1 Overall Aims

The core aims of this project are to gain a deeper understanding of procedural techniques due to it being a complex and interesting field, produce a tool that simplified world creation for fantasy tabletop rpgs which will generate a world map which will contain biomes and continents with tools that will assist in managing the world. This will allow for the game master to focus on telling a story than creating a detailed world. They will still be able to have an input as there are tools which will allow placement of their own settlements. Which will allow the application to be useful in both planning and at the table (during gameplay).

1.2 Objectives

Procedurally generate terrain which will include biomes and continents which will be used for a 3D map for a fantasy world.

Create tools that interact with a generated map to transform the map from just a generated image to a usable and interactable environment. Additionally, the user could add settlements, measure distances between two points, keep track of important characters at the touch of a button.

* Research approaches to biome generation and the corresponding algorithms
* Chose tools based on existing libraries for chosen algorithm
* Start planning implementation based on research
* Implement continents
* Implement terrain
* Implement biomes

Usability tools which will not affect generation but make the generation have value

* Add the ability for the user to add custom labels to the map so they can define their own cites etc in the world.
* Include measurement tools so user can quickly and easily find out the distance between two objects or places within the world
* Add a “party”/ “important person” token so it’s possible to keep track of where certain individuals are in the world.

Stretch goals

* Enforce biome adjacency rules; no snow directly adjacent to tropical forests
* Define crossings for bodies of water. E.g. add bridges or a boat.

2 Description of research/prototyping completed

Research that has been completed so far consists of a literature review which included looking at a wide range of procedural terrain where a lot of new techniques and concepts were discovered for example use of erosion was a completely new concept which is an interesting way to develop natural looking terrain as it follows a natural process (See Appendix C).

2.1 Discussion of potential solutions - Algorithms/approaches

2.1.1 Tectonics

Tectonics is an approach that’s main principle is to simulate tectonic plates moving to create boundary’s mostly in the form of coasts and mountains.

This works by creating the plates and giving them a vector with a direction and distance, and when two plates collide they will form a mountain due to the convergent nature of the plates and the empty spaces formed by the plates diverging will form coasts. (Maddox, 2018)

Strengths

* Creates realistic mountain ranges by simulating tectonic plates colliding
* Movements of plates naturally form coasts and borders

Weaknesses

* For more accurate tectonics’ simulation, it becomes computationally complex

2.1.2 Erosion

Erosion is an approach which is done after an initial terrain generation has been completed, this can be based of terrain generated via a Voronoi style, a Perlin noise style or any other similar approaches to generation. Where erosion algorithms are applied afterwards which makes the terrain appear weathered.

This approach makes the terrain look detailed and naturally aged in appearance. Although this comes at quite the performance cost (Olsen, 2004) as the erosion algorithms are computationally complex and can take up to a couple of seconds to complete depending on the size of the data set (Olsen, 2004) and they need to be run at least 50 to 100 times to really get a suitable appearance (Olsen, 2004). Which will also introduce a lot more complexity to the terrain mesh making it more challenging to render, which will negatively affect runtime viewing.

2.1.3 Hexagonal tile-based generation

A generation approach where each terrain entity is encapsulated inside of a hexagon for example a mountain would exist in one hex in the grid. Which is a very different approach to the other generation methods that have been looked at for this project. This approach can be broken down into place hex’s in a grid, use noise to decide what is on the hex (landon912, 2011). This approach can fulfil all this project’s criterion with the exclusion of the visual representation. As its visual style is that of a civilization game (Firaxis Games, 2016).

Strengths

* Allows for integration into a hex based game with the utmost simplicity

Weaknesses

* Not visually appropriate for this project

2.1.4 Voronoi Diagrams

A Voronoi diagram is a way to create polygons based off an inputted set of points, which is created by passing the points into the fortunes algorithm (Fortune, 1986) The fortunes algorithm works by moving a line downwards over the set of points, when the line intersects with a point an arc will be created equal distance between the line and the point (it will always remain equal between the two) this happens each time the line intersects with a point. If three arcs intersect this means that a hard edge of a polygon can be created from the point of intersection to the closest other hard edge. This continues till there are no more points (Heunis, 2018).

There are a lot of similar project which used this approach to create their own generators all based off Amit Patel’s article (Patel, 2010)covering this approach a couple of which are (reference all those 3 projects/articles)

Strengths

* Searching a Voronoi diagram is easy.
* Creates as a titled surface efficiently as it can be as fast as O(n log n) time (Anon., 2017)

Weaknesses

* Can only be calculated based off two axes

This approach has been selected for this project due to how many other projects have achieved similar results to the desire generation goal, lots of which have articles explaining their approach which will be incredibly helpful when issues occur in the project. There are also numerous libraries for the Fortunes algorithm which will be helpful.

2.2 Discussion of tools and technologies – Criteria 4 25% - 250 words

This project will use Unity as its engine, as that will allow the project to focus on the procedural generation and not waste time developing a custom environment using an API to render everything. As for picking Unity specifically, (as similar projects have been completed with this approach it seemed most appropriate to use) this is due to it being the engine I have the most experience in, in turn meaning time will not be wasted trying to learn a brand-new tool

(See Appendix E).

Having spent a week trying to learn the fortunes algorithm (to be able to implement it without a library), a greater understanding of how the algorithm functions was developed. As a result, pseudo code based on this understanding was created (See Appendix F). However, due to this new understanding, it was clear that it would be too challenging to implement without hindering the project’s progression.

Desired features of Voronoi library

* Creation of Voronoi diagrams
* Creation of Voronoi diagram onto a mesh
* Delaunay implemented

The project will use Jceipek’s library for Voronoi and Delaunay triangulations(jceipek, 2016)as the library to provide both the fortunes algorithm to create Voronoi diagrams and Delaunay triangulation. This is due to it being the easiest to understand, which is because its code being structured in the clearest way and the closest to other examples of the fortune algorithm. While also being feature complete (See Appendix G).

3 Project Specification

This project will be:

An application capable of generating 3D maps for a fantasy world,

The 3D map will have clear biomes and visible terrain,

The user will have the ability to alter parameters prior to generation,

The user will have the ability to save and load generated maps,

The application will have user tools which will allow them to create labels for any location. Additionally, they will be able to select an icon to go with the text which can be placed anywhere on the map, which is intended to be used to define where settlements, dungeons or any other important landmark may be,

Another tool the user will have is the ability to place moveable tokens on the map which are intended to be used to track the location on important character and where they are in the world.

4 Discussion of software dev methodology

The project has been broken down into time boxes (see Gantt chart) in which each will be begun by doing any necessary flowcharts/pseudo code to break down the technical side of said time box, then begin implementation. Once the implementation is complete the time box will be reviewed, and the remaining time boxes will be altered to reflect any changes that need to be made to the process.

All the time boxes have been added to a Trello board, the ones which are to be tackled on that week will be moved into a separate list in order.

Progress will be re-evaluated at the end of every time box and at the end of every week to allow for constant tracking of how well the project is progressing and alter scope if necessary based on progress.

All foreseeable risks have already had backup plans put into place (See Appendix H). Any unexpected risks will have to be handled upon them happening which could result in delays in completing features or the project.

5 Discussion of project management tools and metrics

This project will follow the Timebox management methodology.

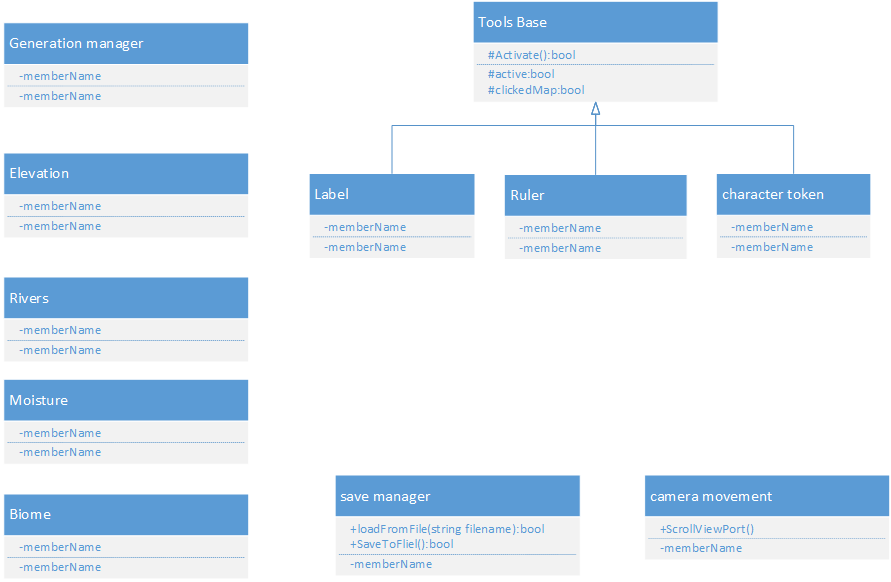
Two Gantt charts have been produced for the initial plan for the project through to completion (See Appendix I). Where one estimates time based on only minor issues occurring the project, while the second has doubled the estimated time to better allow for issues and difficulties that will appear to be handled, even with this larger estimate, the project will still be completed on time.

6 Resource implications

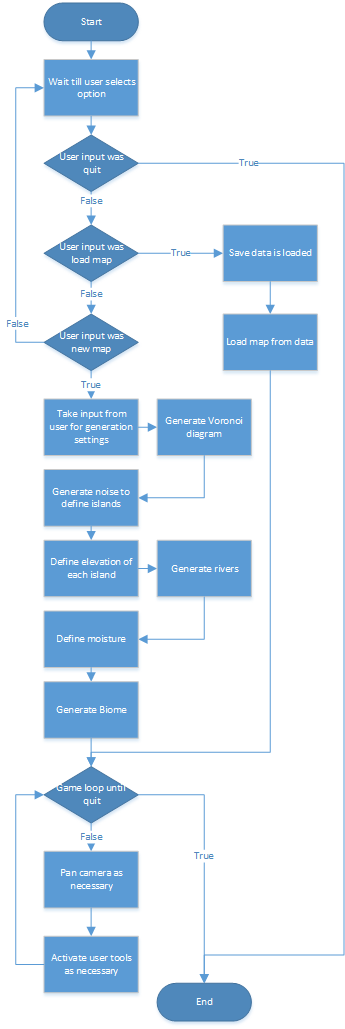
This project will require testing by a user group that make up the target audience, as it is a product designed to be used by people to assist with their table top gaming. People at Solent’s D20 tabletop gaming society will be approached and asked if they would like to provide feedback after using the application.

7 High level overview of classes

This is the high-level class diagram for the project. All tools will inherit from tools base, so they can have all the same core components so that all tool will have the same core functions. Generation has been broken down into several separate classes to ensure that each class will have high cohesion as they will all exist inside the generation manager and will have their functions one after another, each step broke down into its own functions to make for easier debugging. The save manager will be passed data as the generation process happens to ensure that the same world can be re-generated on loading. Camera movement just to allow the users to see different parts of the map.



7 High level flow diagrams and pseudocode

This is the current plan of how the application will flow once running. It’ll wait at the main menu until an option is selected.

Implementation – 5500ish

Other things the section May contain

* How the project progressed: Did it stay on track?
* Changes made (if any) to the schedule, why and when did they happen?
* What was learned from each milestone?
* Feedback from end-users and/or questionnaires.
* Difficulties encountered during the implementation stage.
* Implementations of backup plans.
* Any interesting implementations of tasks.
* Any other events of note that occurred during the development cycle of the project.

Introductory page

* Bridging the time gap since progress report????
* WHAT IS THIS SECTION MARK????

Formalized logbook -550ish per timebox

* Formalized and professional version of logbook
* Discussion of implementation chronologically by time box would be logical.

Evaluations, Reflections and Future Development -1500ish

Evaluations - A critical discussion of the project:

• Does it meet the desired aims and objects defined in the project definition?

• What processes have been performed to verify this?

• Were certain objectives not met? If so, discuss why.

Reflections - A post-mortem discussion of the project as a whole:

• What went right?

• What went wrong?

* + Initial library was bad, had to change 2 days into starting.

• What was learned?

• What could have been done differently?

• If the project were to be repeated, what changes would be made to improve the project?

Future Development - A discussion of the potential continuation of the project:

• What questions has the project raised?

• How could the project be used as a foundation for future development?

Appendices

Appendix A - References

Azgaar, 2017. *Biomes generation and rendering.* [Online]   
Available at: https://azgaar.wordpress.com/2017/06/30/biomes-generation-and-rendering/   
[Accessed 10 2 2019].

coast, W. o. t., 2014. *Dungeons and Dragons.* [Online]   
Available at: http://dnd.wizards.com/  
[Accessed 1 2 2019].

Fortune, S. J., 1986. A sweepline algorithm for Voronoi diagrams. In: *SCG '86 Proceedings of the second annual symposium on Computational geometry.* New York: AT&T Bell Laboratories, pp. 313-322.

jceipek, 2016. *Unity-delaunay.* [Online]   
Available at: https://github.com/jceipek/Unity-delaunay  
[Accessed 12 2 2019].

landon912, 2011. *[OPEN SOURCE]Procedural Hexagon Terrain.* [Online]   
Available at: https://forum.unity.com/threads/open-source-procedural-hexagon-terrain.233296/  
[Accessed 20 2 2019].

Maddox, S., 2018. *for h in hexes.* [Online]   
Available at: https://forhinhexes.blogspot.com/2018/04/motivation.html  
[Accessed 10 2 2019].

Olsen, J., 2004. *Realtime Procedural Terrain Generation,* s.l.: Department of Mathematics And Computer Science .

Patel, A., 2010. *Polygonal Map Generation for Games.* [Online]   
Available at: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/  
[Accessed 26 1 2019].

Appendix B - Reading list

Beckham, C. & Pal, C., 2017. *A step towards procedural terrain generation with GANs.* [Online]   
Available at: https://arxiv.org/abs/1707.03383v1  
[Accessed 15 2 2019].

Bradley, A., 2018. *Devs weigh in on the best ways to use (but not abuse) procedural generation.* [Online]   
Available at: https://www.gamasutra.com/view/news/315400/Devs\_weigh\_in\_on\_the\_best\_ways\_to\_use\_but\_not\_abuse\_procedural\_generation.php   
[Accessed 26 1 2019].

Compton, K., 2016. *So you want to build a generator….* [Online]   
Available at: http://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator  
[Accessed 26 1 2019].

Appendix C - Literature review

<http://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator> overview on procedural generation which describes a good process on how to break down the needs of the generator and specifically want is desired from the generator. This is not specifically linked to the project, just the general field of generative content as a whole but it can help with the initial breaking down on needs and requirements.

<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/>   
This article covers using Voronoi diagrams to create terrain and uses whittaker diagrams to define biomes. This will be a very useful article if this approach is taken as it goes through the entire process to generate terrain as the end result has all the generative features this project requires.

ISBN 0-89791-194-6  
<https://dl.acm.org/citation.cfm?id=10549> the original publication of the Fortune's algorithm which is used to create Voronoi diagrams. Very difficult to understand due to its’ pure technical approach. It is very relevant as the original source but is not suitable as it’s very challenging to follow and newer sources pertaining to the fortune’s algorithm explain the process much better.

<http://drp.math.umd.edu/Project-Slides/DRP_Presentation-Summer2016.pdf> This presentation briefly covers the fortunes algorithm but does not do a good job of breaking it down a into easily understandable format.

<http://www.rigi.cs.uvic.ca/downloads/papers/pdf/cg.pdf>

This presentation has a reasonable explanation of the fortunes algorithm but portions of it are very difficult to understand due to it being in a mathematic formula using symbols that do not have a key explaining what they mean.

<https://jacquesh.github.io/post/fortunes-algorithm/>

A well detailed article explaining how the fortunes algorithm functions step by step of each stage with interactive demos. This was the main source that led to the understanding of the fortunes algorithm that I now have.

<https://jacquesheunis.com/post/fortunes-algorithm-implementation/> part 2 focus on implementation

<https://azgaar.wordpress.com/2017/06/30/biomes-generation-and-rendering/> This article covers using Whittater diagrams to generate biomes on a polygonal map, also brings up Holdridge life zones could be used as an alternative <https://en.wikipedia.org/wiki/Holdridge_life_zones> It also covers an approach for rendering the biomes which certainly could be helpful

<https://arxiv.org/abs/1707.03383>

A paper covering terrain generation which focuses on modelling based off of satellites imagery from NASA, an interesting paper which could allow for more realism in the generator, although it is unlikely to prove useful due to the focus being on creating fantasy maps.

<https://dl.acm.org/citation.cfm?id=1814259>    
Towards multiobjective procedural map generation

This paper focuses on generating game maps for strategy games which are required to be balanced so it focuses on finding pairs of locations in which objectives could be placed. This is not particularly useful to this project as world generation shouldn’t be balanced.

<https://pdfs.semanticscholar.org/5961/c577478f21707dad53905362e0ec4e6ec644.pdf>

A paper on real-time procedural terrain generation which focuses on using near real-time erosion to sculpt the landscape, where it covers a way to generate terrain using fractal noise and then ways to implement two different erosion methods onto said system, thermal erosion and hydraulic erosion. This is a useful paper and is an approach which could work for the terrain generation portion of the project.

<http://slideplayer.com/slide/3447433/12/images/14/Robert+Whittaker,+Cornell+Uni..jpg>

A Whittaker diagram which describes how temperature and moisture levels define biomes. A good basis to base biome generation off of, will be useful towards this project.

<http://www.jgallant.com/procedurally-generating-wrapping-world-maps-in-unity-csharp-part-1/> A tutorial series which discusses a way to implement biomes/ world maps which could be very helpful, as the end results are similar to this project’s end result. It also focuses on wrapping the map onto a sphere which will is not within the projects current scope.

<https://forum.unity.com/threads/open-source-procedural-hexagon-terrain.233296/> An in-depth approach at how to generate hexagonal terrain in which mountains will be contained in a hex tile. Resulting in a visual appearance not idea of this project as this tutorial’s goal is to generate civilization 5 like maps. Still it is a useful resource which would fulfil all of the projects other criteria.

<https://forhinhexes.blogspot.com/2018/04/motivation.html> This has a very similar end goal as to this project but represents the map differently from how is currently desired for this project. Which uses tectonics to generate its islands and mountains. Not too useful a source due to its different visual representation.

<https://www.gamasutra.com/view/news/315400/Devs_weigh_in_on_the_best_ways_to_use_but_not_abuse_procedural_generation.php>

An article which focuses on good ways to use procedural generation a very interesting article about strengths and weaknesses of generation, it also states that generative content can be created in any tool which is a refreshing view. Outside of the original design stages this will not be a very useful article for the project.

Appendix D – reference image



Figure 1 map from skyrim

Appendix E - Discussion of tools and tech

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Engine/API | Language used | Experience | Procedural content developed previously (by myself) | Strengths | Weaknesses |
| Unity | C# | 10+ games | <https://thedarkmagi.itch.io/procedural-world-generation>, others which are less relevant (2d generation) | Easy to use, rendering is handled automatically, very familiar with, good documentation easy to understand and use | 65k verts max per generated mesh. By default, for mobile compatibility (could be change to 4billion verts at the cost of compatibility)  <https://docs.unity3d.com/ScriptReference/Mesh-indexFormat.html> |
| Unreal | C++/Blueprint | 3 games with blueprint | Chunk spawning generation | Easy to use, very good rendering capabilities (shit looks good), designed for 3D | No experience using C++ in unreal.  Same vertex issue as above due to mobile index buffers being 16bit. Quite bloated in size as a lot of features which might not be used are included by default |
| Godot | GDScript, visual scripting, C# and C++ <http://docs.godotengine.org/en/3.0/about/faq.html> | Never used | None | Open source with no royalties, completely free to use | For the best experience it requires learning a new language |
| Game maker | GameMaker Language (GML) | Never used | None | Simple and easy to use | Designed for 2D games, costs money to use, requires learning a new language |
| Heaps.io | Haxel | Never used | None | Developed for multi-platform release, entirely free, open source | Requires learning a new language |
| DX11 API | C++ | One assignment | None | Can result in a lot better performance when done correctly, As it is much lower level | Time consuming to setup to get a competent base |

Appendix F - Fortunes Algorithm pseudo code

Setup event queue  (Q)

while Q is NOT empty

Point = lowest event from Q

if point is a site event

Add new site to beachline(point)

else (it is an edge intersection)

remove Squeezed cell from beachline(point)

end while

resolve any unfinished cells by making the edges go to the end of surface.

Add new site to beachline(event point)

find arc above point AF

new arc from point AN

split arc into two pieces AL and AR

   create two edges inside the new arc facing in opposite directions EL and ER

remove AF from beachline.

insert AL, EL, AN, ER, AR in this order into the beachline

Check for any circle events caused by this. (point's AL and AR)

remove squeezed cell from beachline(event point)  / circle event

find edge to the left and right of point EL and ER

if EL || ER  == null

break

if EL does not intersect || ER does not intersect

break

DO the circle event thing??

Giving interesting edges end points at the intersection.

Create a new edge facing downwards from the intersection point

Remove arc and edges from beachline.

Add new edge to beachline

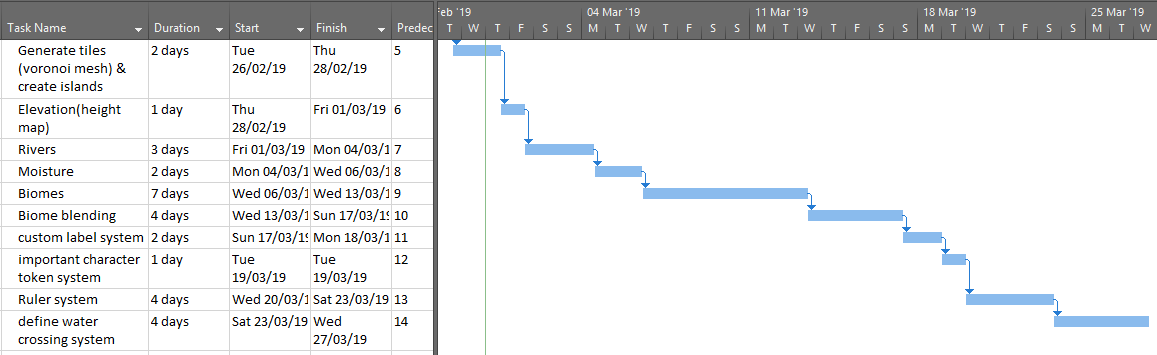
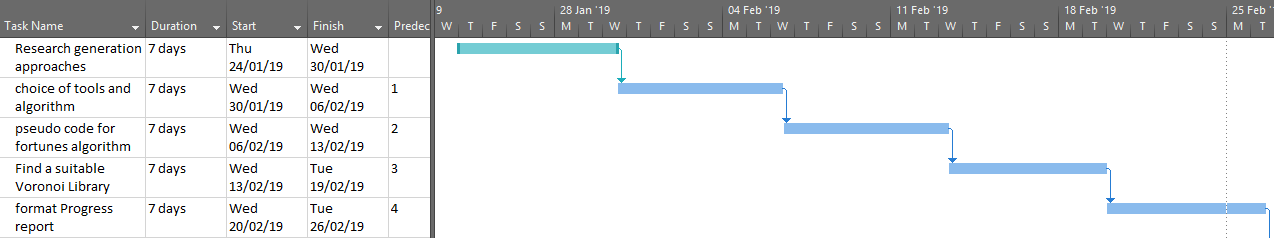
Appendix G - Library Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Feature complete | Compatible | Easy to use/understand |
| <https://github.com/jceipek/Unity-delaunay> | Yes | Apart from demo project. yes | Seems it? |
| <https://github.com/Ranguna/Triangle-NET-Unity-Port> | No Delaunay triangulation | Yes | Difficult to understand how to use the library. |
| <https://github.com/eppz/Triangle.NET> | No |  |  |
| <https://github.com/PixelsForGlory/VoronoiDiagram> | No Delaunay triangulation | Unable to know due to not being able to install. Effectively incompatible. | The installation process is unclear making it unusable |
| <https://github.com/OskarSigvardsson/unity-delaunay> | Delaunay and Voronoi. Voronoi is generated from Delaunay, I don’t know if that convenient based on my current approach | Compatible |  |
| <https://github.com/akopetsch/triangle-unity> |  |  |  |

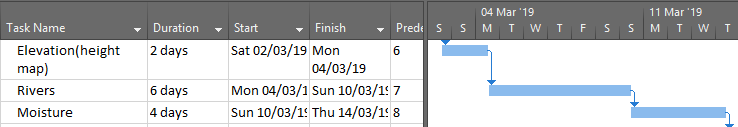
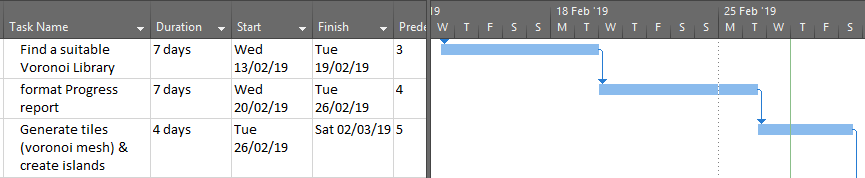
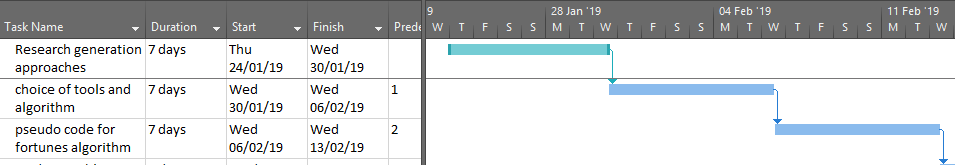
Appendix H – Risk Analysis

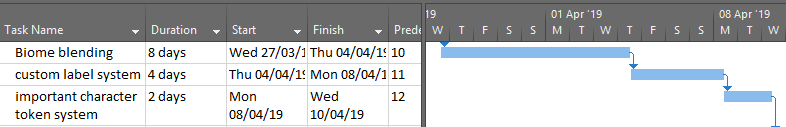
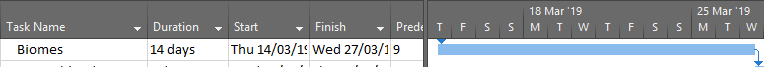
|  |  |  |
| --- | --- | --- |
| Risk | Likelihood | Backup plan |
| Difficulties implementing chosen algorithm | high | Use pre-existing libraries to allow the project to continue |
| Engine having unknown limitations | Medium | Features may have to be removed if a solution to bypass the limitation cannot be found |
| Libraries being incompatible with recent versions of unity | medium/high | Find another library or reconsider use approach/engine to use |
| Libraries not having all required features | Medium | Find other ones. |
| Core objectives taking too long to implement | Medium/high | Removal/scaling down of less important core features to fit within timescale correctly |
| Home PC breaks irreparably | Medium/low | Use university computers to work from as all progress is backed up using source control very little work should be lost. |
| Project data loss | Low | At the end of a day create backups of project locally, in the cloud (google drive) and on a memory stick encase Github were to fail. |

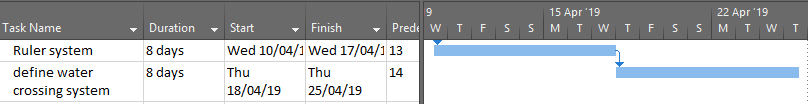
Appendix I - Gantt Charts

Estimated duration assuming only small issues occur

Estimated duration doubled to allow for larger issues







Appendix J - Work Log

28/01/19

* Research into multiple different kinds of terrain generation techniques

05/02/19

* Research into different tool that could be used
* More research into algorithms for generation

08/02/19

* Being researching into how the fortunes algorithm functions

09/02/19

* Further research into the fortunes algorithm
* Begin pseudo code for the fortunes algorithm

12/02/19

* Finished pseudo code for fortunes algorithm

16/02/19

* Research into different libraries for Voronoi diagrams

25/02/19

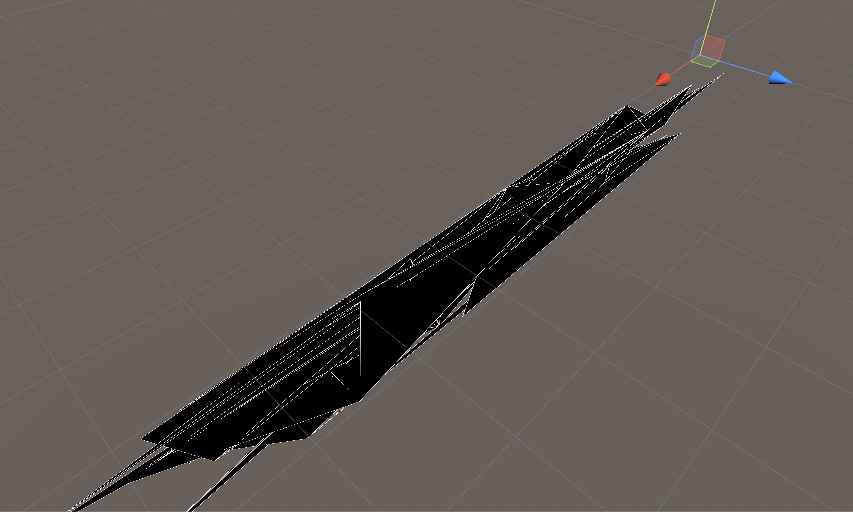
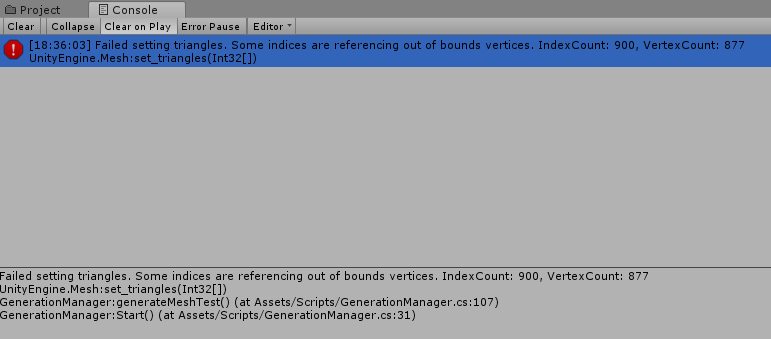
* Gantt charts created both long and short estimates

26/02/19

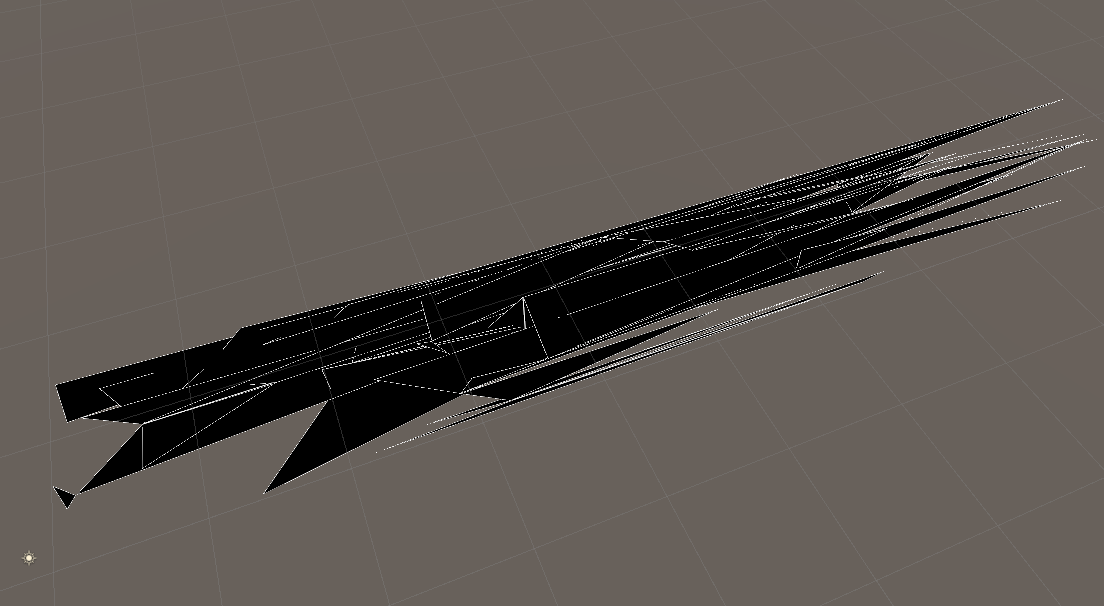
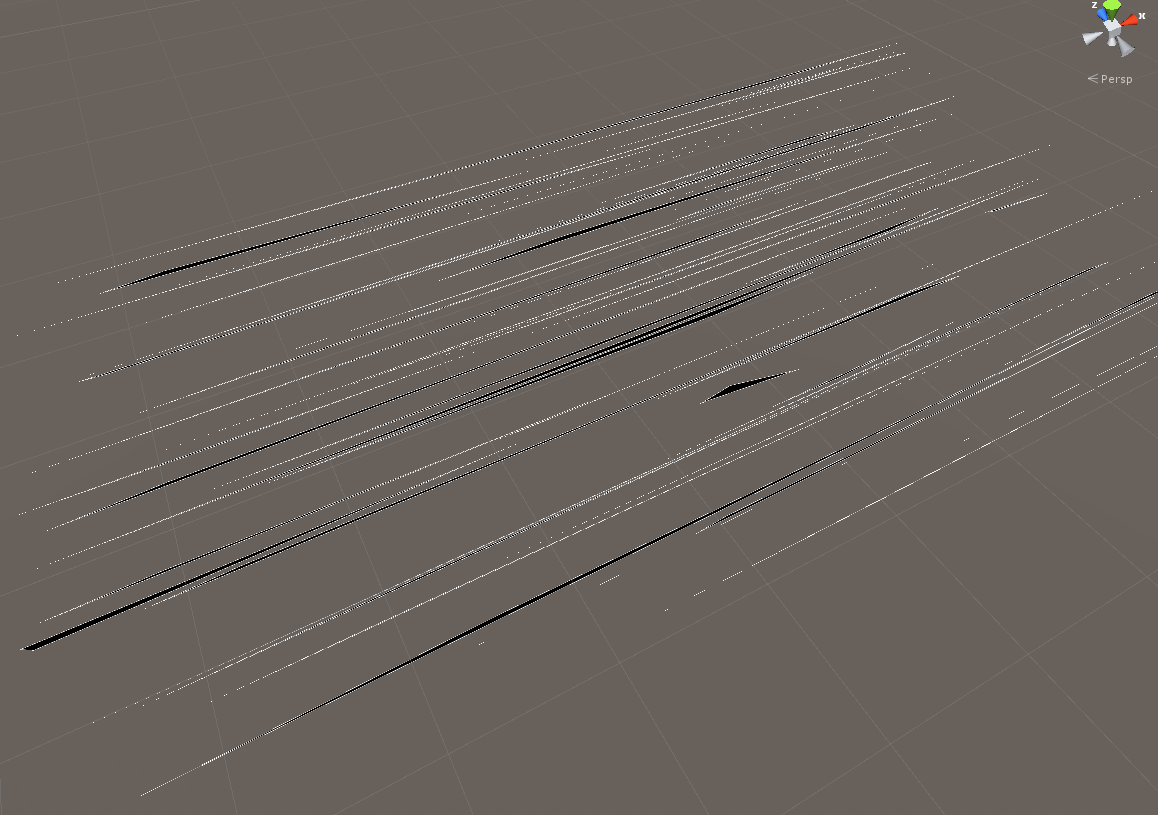
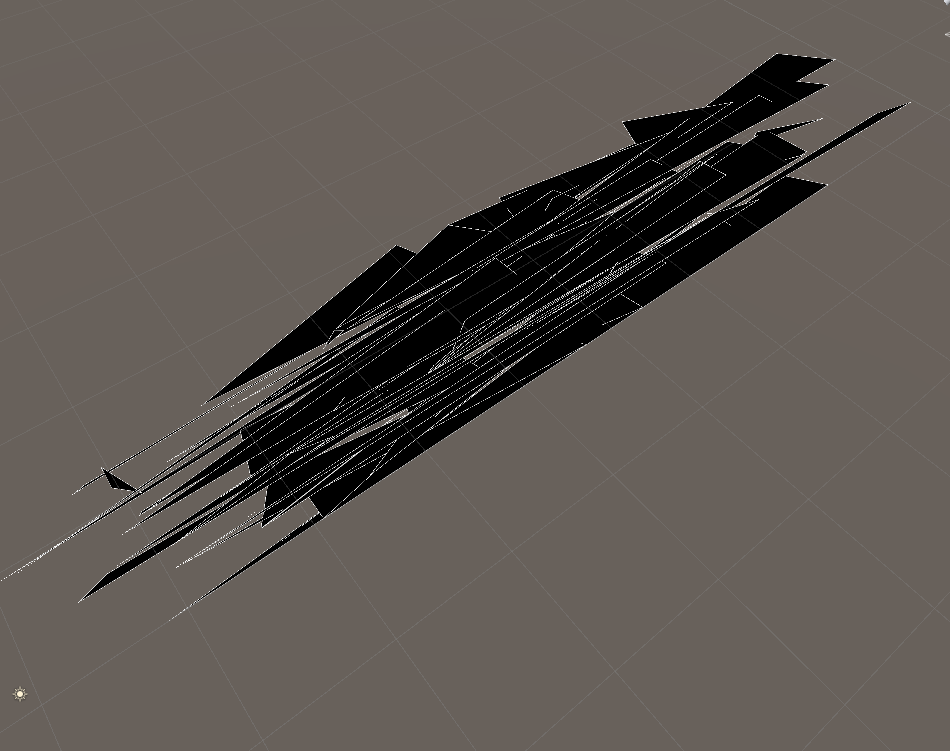
Focus on improving progress report

* Filling in more details in literature review
* Rewording tools section so it’s no longer in first person
* Wrote up about erosion style generation

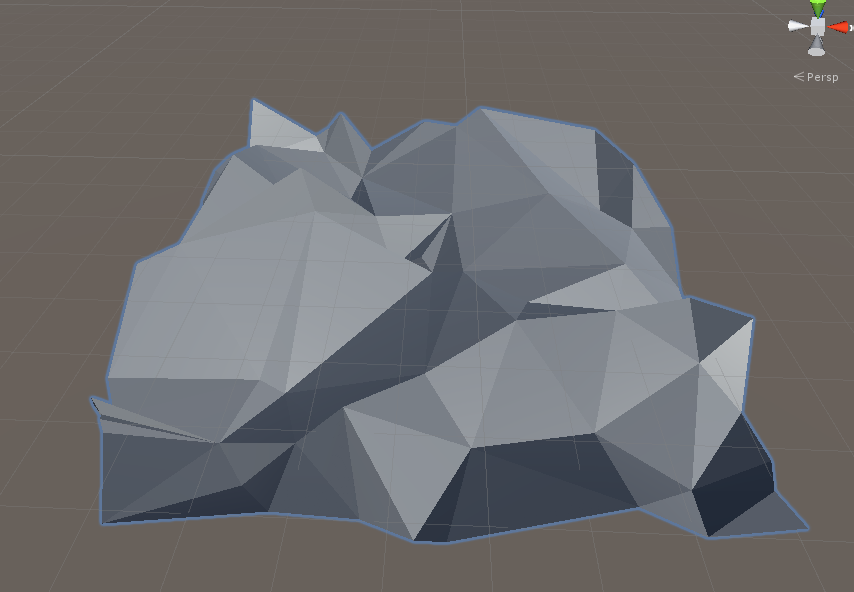
02/03/19

* Setting up Final report structure
* Being on generating a mesh from voronoi diagram
* (first mesh actually rendered at all)
* delancy list is not as long as the n vertices when using sqMagnitude

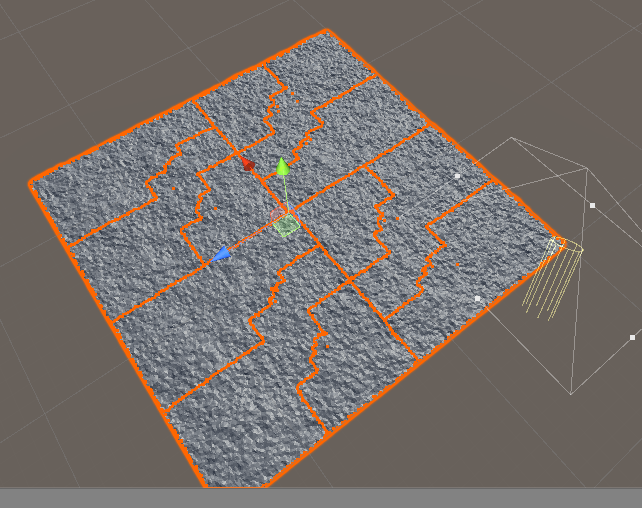
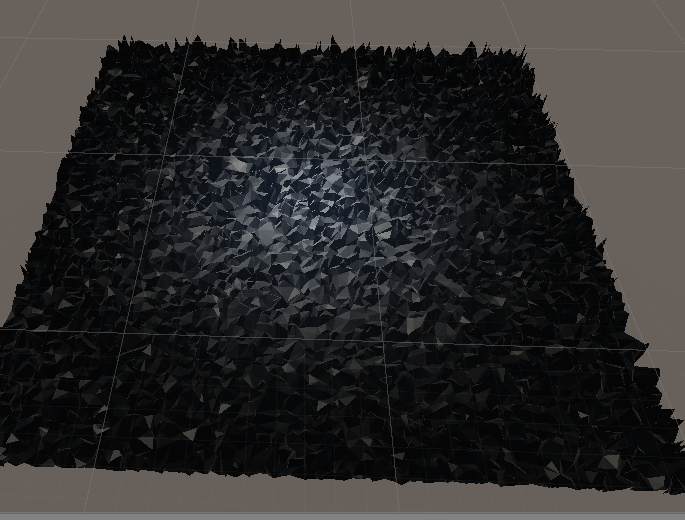
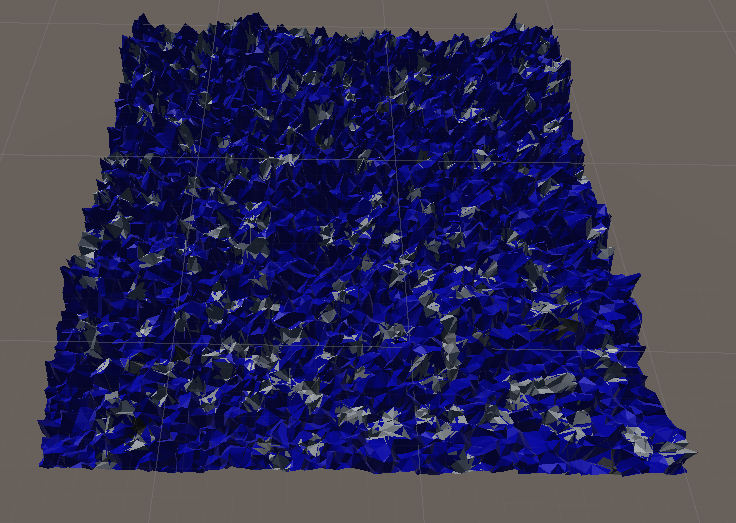
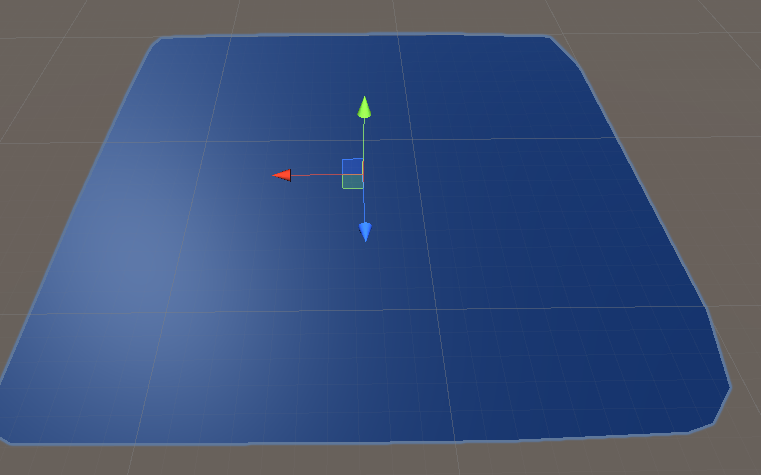
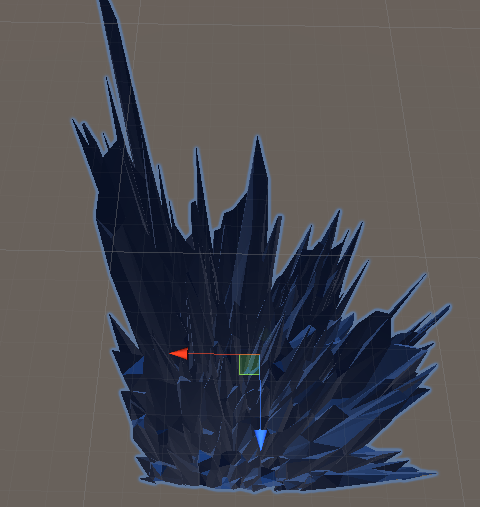
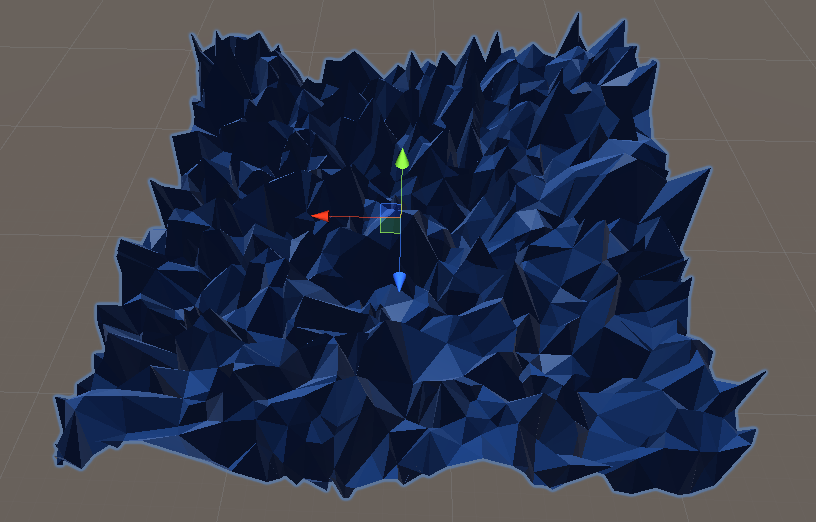
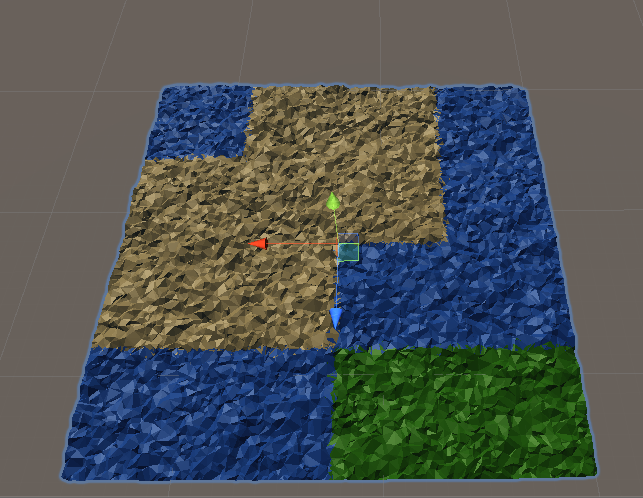
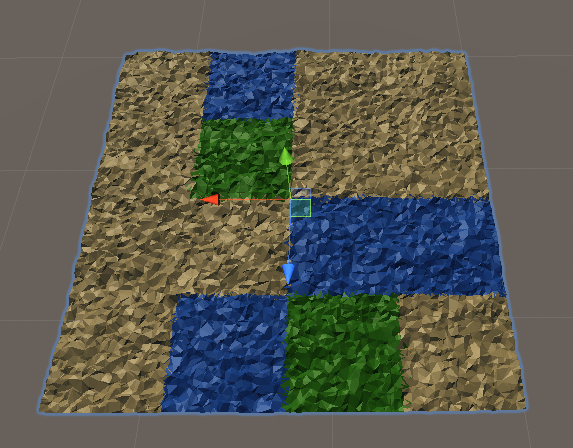
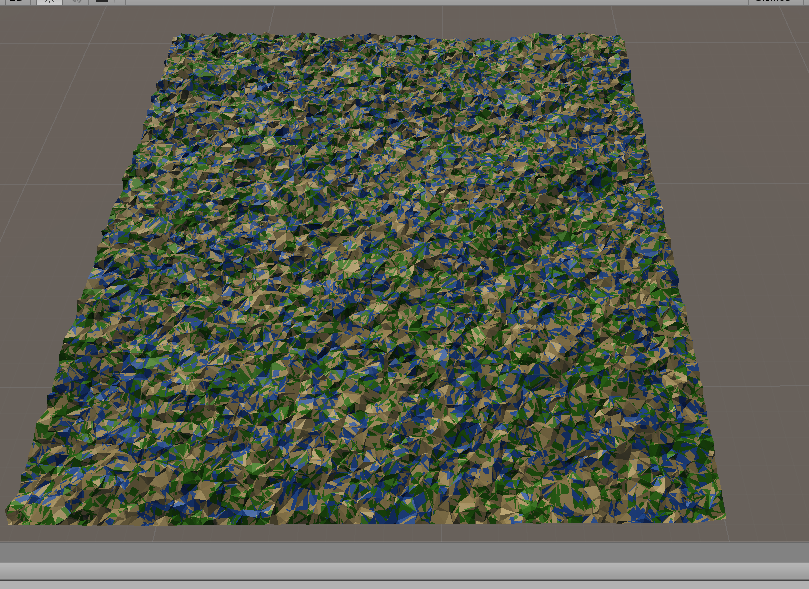
03/03/19

* Continuation on generating mesh
* 
* 
* Using Voronoi sites as verts
* 
* Using points from Voronoi edges.
* The core issue here is that no matter what data is being inputted the mesh is not being given a proper triangle list as calculating which points of make up each triangle wasn’t easy nor could I understand how to calculate it outside of A grid structure of vertices. Which voronoi diagrams aren’t
* Added new library (don’t understand how to use it yet.) as I’m having difficulties correctly generating a mesh with my current library choice

04/03/19

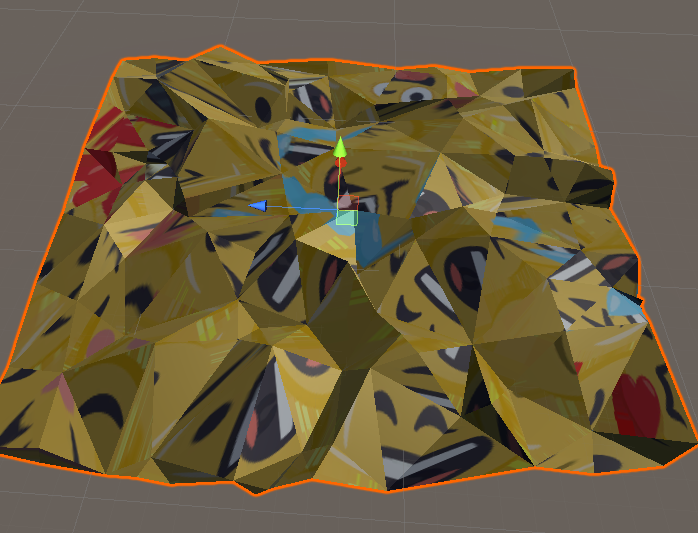
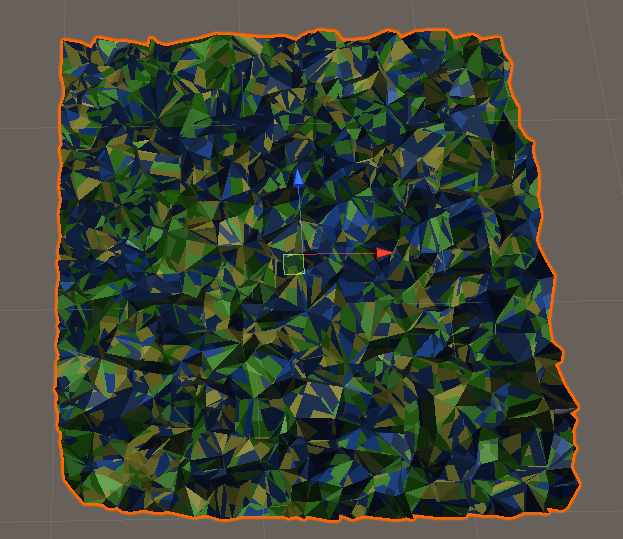
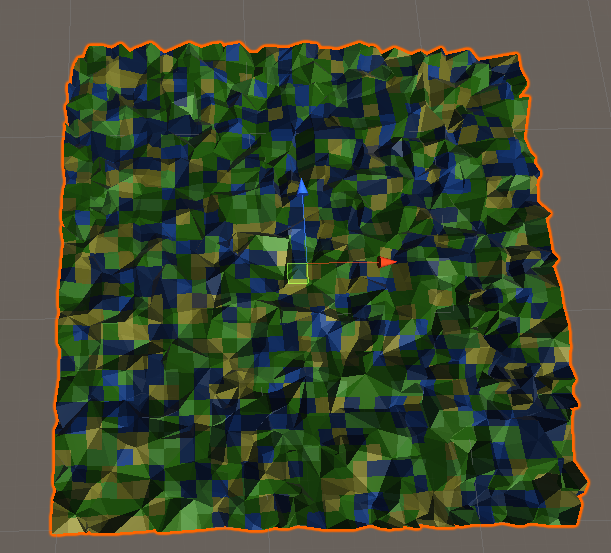
* Swapped to using triangle.net as it’s got actual mesh integration
* Followed a tutorial on how to get triangle.net functioning. Using Delaunay instead of Voronoi <https://straypixels.net/delaunay-triangulation-terrain/>
* 

05/03/19

* 
* This approach generates chunks based on a max tri count which maps them appear very interestingly. With no viable seams between chunks
* 
* Correctly generating uv’s based on triangle position.
* 
* Perlin noise applied to mesh as texture? ( I think, it looks like it ) only 2 colours from the look of it. Also height map is going to have been done at this point aswell.
* 
* Trying to just replace colours under a certain value seems ineffective. :/
* 
* Normalised all values to 0 like a clown….
* 
* Improved but still very incorrect ( all numbers are sub 0)
* 
* Had a / when it should have been a - normalisation fixed colours not so much :/
* 
* Tecture is interesting???
* 
* Texture is scaling crazy large/ it’s actually just a 4x4 resolution texture
* 
* I needed to calculate the size of the texture correctly. To get a consistently correct size I had to sqrt the length of the colour map.
* 
* Colours might still be being applied incorrectly.

07/03/19



* Textures are now makes sense based on their height. The issue was elevation was being generated twice and not causing the texture to line up with the mesh.
* NOT THE FIX this just appeared to work from sheer luck.
* There is a problem with the UV calculations as other textures are also unable to render correctly
* 
* 
* Attempts at using bounds to normalise the vertex values to become uvs
* 
* A very different result. UV’s are now being done in reverse order like the verts and are being divided by the bounds of the mesh to give them the correct UV. It looks like colours are still being set incorrectly but still a step in the correct direction.
* I’ve realised why the colours from the texture don’t line up with the mesh, it’s because all the pixels are being set in order based on the height of a vertex, but unlike the texture the vertexes are not linearly spaced. Meaning that this approach will never work with this level of sampling heights.
* Possible solutions ( from brain, not researched ):
* Either sample heights at uniform points once the mesh has been all setup to be passed into a grid
* Or use vertex painting to colour each of the vertexes and have colours transition between vertexes to colour the triangle