[Date]

Oliver Mills

q12339911

[Company address]

Final Major project

Progress report

Contents

[Background – part of 1, a portion of 15% -150 words curr 125words 2](#_Toc1920541)

[Overall Aims – part of 1, a portion of 15% - 150 words curr 46words 2](#_Toc1920542)

[Objectives – part of 1, a portion of 15% - 150 words curr 128 words 2](#_Toc1920543)

[Description of research/prototyping completed – criteria 2, 20% - 600 words 3](#_Toc1920544)

[Project Specification – Criteria 3, 15% 450 words 3](#_Toc1920545)

[Discussion of potential solutions - Algorithms/approaches –Part of Criteria 4 25% - 250 words 3](#_Toc1920546)

[Discussion of tools and technologies – Part of Criteria 4 25% - 250 words 4](#_Toc1920547)

[Discussion of software dev methodology – Part of Criteria 5 25% - 375 words 5](#_Toc1920548)

[Discussion of project management tools and metrics – Part of Criteria 5 25% - 375 words 5](#_Toc1920549)

[Resource implications 6](#_Toc1920550)

[High level overview of classes that may be required –Part of Criteria 4 25% - 125 words 6](#_Toc1920551)

[High level flow diagrams and pseudocode –Part of Criteria 4 25% - 125 words 6](#_Toc1920552)

[Appendices 6](#_Toc1920553)

[Reference List and Reading list – part of 2, a portion of 20% 6](#_Toc1920554)

[Literature review – part of 2, a portion of 20% 6](#_Toc1920555)

[Discussion of tools and tech 8](#_Toc1920556)

[Fortunes Algorithm pseudo code 10](#_Toc1920557)

[Discussion of software Dev methodology 11](#_Toc1920558)

Background – Criteria 1, a portion of 15% -150 words curr 125words

This project will focus on generating world maps to simplify and streamline world creation for table top roleplaying games(rpgs) with a focus on fantasy setting similar to ones used in Dungeons and Dragons(D&D) (REFERENCE?). This map will be procedurally generated which will have terrain and biomes, with stretch goals of implementing resources linked to the environment and generating settlements based on said resources. This map will be presented similar to Skyrim’s world map (fig 1) but using a much lower poly style and simplified representation of terrain types as such foliage would not be included.

This project is being created to gain a deeper understanding of procedural world building techniques, which will be technically challenging project thus it is a good fit for the course.

Overall Aims – Criteria 1, a portion of 15% - 150 words curr 46words

* Gain a deeper understanding of procedural techniques
* Simplify world creation for fantasy table top rpgs
* To create a tool that will assist in managing the created world making it useable at the table and not just away from it.
* To generate a world map with biomes

Objectives – Criteria 1, a portion of 15% - 150 words curr 128 words

* 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

Stretch goals

* Enforce biome adjacency rules; no snow directly adjacent to tropical forests

Usability tools which don’t affect generation at all 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 find out how far away something is for another
* Add a “party”/”important person” token so it’s possible to keep track of where certain individuals are in the world.
* Define crossings for bodies of water. Eg add bridges or a boat?

Description of research/prototyping completed – criteria 2, 20% - 600 words

* Section on research/prototyping done at this point

AAAAAAAAAAAAAA fill this out you dumbass

Fda

* Research into different types of terrain generation
* Research into map generation types
* Research into fortunes algorithm - shits hard don’t attempt this

Project Specification – Criteria 3, 15% 450 words

* Fully detailed description of the “product” use short clear descriptions with measurable values
* The specification can be used for guidance in determining potential solutions, as well as a basis for the planning and breaking down tasks.
* Wide range of relevant referenced research shown with excellent analysis
* Uses analysis to derive a clear, concise and unambiguous specification based on measurable values.
* An application capable of generating 3D maps for a fantasy world
* Which will have biomes and visible terrain
* Ability to alter parameters prior to generation
* Ability to save and load generated maps.

Discussion of potential solutions - Algorithms/approaches – Criteria 4 25% - 250 words

* Tectonics
  + <https://undiscoveredworlds.blogspot.com/2019/02/basic-continents.html?m=0> use of tectonics to form mountain ranges
  + Strengths
    - Creates realistic mountain ranges by simulating tectonic plates colliding
  + This approach doesn’t use real tectonics at they’ve very complex, they simply created blobs for continents. Doesn’t explain how mountain ranges are completed.
* Voronoi
  + <http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/> The original source which introduced me to the concept, which breaks down an approach of how to generate a map using this methodology. This involves the use of whittaker diagrams which are used to model biomes.
  + In my research I found many similar projects that all used this article ^ as the basis, <https://mewo2.com/notes/terrain/>, <https://azgaar.wordpress.com/2017/03/30/first-post/> which is based of prior link which is in turn based off of  ^ above, same as prior <https://heredragonsabound.blogspot.com/2016/10/welcome.html>
  + Reason chosen
    - Many great examples for how it works
    - Many other projects that have reached my desired result via this method
    - Libraries exist for algorithms I want to implement if I were to fail implementation
  + <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 math existence. ( no pictures :( )
* Hex based generation
  + <https://forhinhexes.blogspot.com/2018/04/motivation.html> This has a very similar end goal as my project but represents the map differently from how I want.
  + Desired generation to be less rational less “smooth” as they believe it will represent life better due to its unpredictable nature. I believe their dislike for seeds and random numbers is due to their lack understanding of how they impact generation, their issues are due to how the data is handled not the seeds themselves. As the data could be handled to generate those irregularities by adding a small chance for them to appear at the point that element is generated.
  + Also using tectonics
    - No interested in tectonics themselves but are interested in the boundaries that are the result of them, convergent and divergent which are used to generate mountains and coastlines respectively.

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

This project will use Unity as it’s 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 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 Discussion of tools n Tech in appendix for more deets

* Libraries available to me? Which languages etc
  + Delaunay libraries
  + <https://github.com/jceipek/Unity-delaunay> c#
  + <https://github.com/SirAnthony/cppdelaunay> c++

After having spent a week trying to learn the fortunes algorithm to be able to implement it without a library I understood how the algorithm functions, created pseudo code based of my understanding (See Appendix) and that it’s too challenging to implement without hindering the projects progression.

* Desired features
  + Creation of Voronoi diagrams
  + Creation of Voronoi diagram onto a mesh
  + Delaunay implemented

The project will use <https://github.com/jceipek/Unity-delaunay> (find a better way to refer to this) 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, due to its code being structured in the clearest way and the closest to other examples of the fortune algorithm. While also being feature complete.

See Discussion of tools n Tech in appendix for more deets

Discussion of software dev methodology – Criteria 5 25% - 375 words

Answer these

* How to do you plan to begin the project?

#By planning the plan setting up timeboxes, filling up a Trello so I will not have to waste much time later setting up plans and I can just continue working with ease

By implementing the chosen library to begin generation

* How will you organize the plan and weekly workload?

Using Trello to store my weekly tasks

* How frequently will you re-evaluate progress compared to the plan?

On a weekly basis

* How will risks be managed?

See Discussion of software dev methodology for deets

Discussion of project management tools and metrics – Criteria 5 25% - 375 words

* Research – 5 days
* Comprehend research and understand algorithms - 4 days
* Generate biomes – 2 - 4 weeks possible breakdown of generation based on current level of research.
  + Generate tiles (with Voronoi)
  + Assign tiles biomes
  + Enforce biome adjacency rules (no snow next to tropical forest etc.)
  + Biome blend/transition tiles
  + Heightmap implemented in relation to biome type (mountains high, sea low)

Stretch goals - 2 weeks

* Assign resources to tiles
* Spawn cities with locations based on resources
* Scale cities based on available resources

Plan mark 2 - skipping research cuz that’s done

Core generation 19 days / 38 days total

* Generate tiles (Voronoi mesh?) & Create islands – 2 days / 4 days
* Elevation (height map?) – 1 day / 2 days
* Rivers – 3 days / 6 days
* Moisture – 2 days / 4 days
* Biomes – 7 days / 14 days
* Biome blending - 4 days / 8 days

Tool Block 11 days / 22 days total

* Custom label system – 2 days / 4 days
* Important character token system (put movable tokens on the map) – 1 day / 2 days
* Ruler system, calculate distance (in miles or meters? Stretch x days walking, x days by horse) – 4 days / 8 days
* System to define crossable points for bodies of water (bridges, boats etc) – 4 days / 8 days

Stretch goals if any time is left

* Assign resources to tiles
* Spawn cities with locations based on resources
* Scale cities based on available resources

Resource implications

Testing

High level overview of classes that may be required – Criteria 4 25% - 125 words

A high-level class diagram with inheritance, composition and aggregation

High level flow diagrams and pseudocode – Criteria 4 25% - 125 words

Approximate indication of the ordering of operations

Appendices

Reference List and Reading list – part of 2, a portion of 20%

Literature review – part of 2, a portion of 20%

<http://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator> overview on proj gen

<http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/> article with an approach I’d like to take.   
This article covers using Voronoi diagrams to create terrain and uses whittaker diagrams to define biomes.

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 fortunes algorithm explain the process much better.

//

Voronoi learnings

<http://drp.math.umd.edu/Project-Slides/DRP_Presentation-Summer2016.pdf> kinda covers stuff

<http://www.rigi.cs.uvic.ca/downloads/papers/pdf/cg.pdf> has BIG Math confuse. Also Algorithm ( seems to explain somewhat well)

<http://www.cs.sfu.ca/~binay/813.2011/Fortune.pdf> appears to go very step by step

<http://people.math.gatech.edu/~randall/Algs07/mount.pdf> A lot of text as lecture notes ( could be useful unchecked)

<https://jacquesh.github.io/post/fortunes-algorithm/> good examples ( seems to explain very well. Might be my key to kingdom hearts. The arc-blade    - actually re-read through and start making pseudo code

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

//

<https://azgaar.wordpress.com/2017/06/30/biomes-generation-and-rendering/> biome generation

<https://arxiv.org/abs/1707.03383> paper on terrain generation

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

<https://pdfs.semanticscholar.org/5961/c577478f21707dad53905362e0ec4e6ec644.pdf> Realtime Procedural Terrain Generation  -- this looks like a very good source

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

<http://www.jgallant.com/procedurally-generating-wrapping-world-maps-in-unity-csharp-part-1/> a way to implement biomes/ world maps    seems like it could be very helpful

<https://forum.unity.com/threads/open-source-procedural-hexagon-terrain.233296/> generate hexagon style terrain like civ

<https://www.gamasutra.com/view/news/315400/Devs_weigh_in_on_the_best_ways_to_use_but_not_abuse_procedural_generation.php> last section talks about how “any tool can be used“

Books:

Procedural Content Generation for Unity Game Development

Procedural Content Generation for C++ Game Development

Procedural Generation in Game Design

<http://pcgbook.com/> a book on procedural generation in games



Figure 1 map from skyrim

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 |

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

|  |  |  |  |
| --- | --- | --- | --- |
| Source | Feature complete | Compatible | Easy to use/understand |
| <https://github.com/jceipek/Unity-delaunay> | Appears so | Apart from demo project. yes | Seems it? |
| <https://github.com/Ranguna/Triangle-NET-Unity-Port> | No Delaunay triangulation I believe? | 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 |  |

Discussion of software Dev methodology

|  |  |  |
| --- | --- | --- |
| 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. |