Conor Creagh

Student No. 13454222

Class: 4BS2 Computer Science pathway

Project Definition Document

Project description:

Procedural (random) map generation in computer games, this project will investigate the techniques used for procedural map generation in computer games, and will develop a random map making tool. The map could be 2D or 3D, could use a game engine if desired, and could use the student’s own artwork or open-source artwork. The project would involve investigation of various techniques (Perlin noise, cellular automata, Voronoi maps etc.). The requirement is to be able to specify various parameters, a random seed, and generate a map which can then be scrolled around to view. There is no requirement for any functioning gameplay. E.g. see <http://pcg.wikidot.com/category-pcg-algorithms>

Aims of the project:

1. To look at the different techniques that are used in the world of procedural map generation such as Perlin noise and its successor simplex noise. This will take up a large portion of the project as there are many techniques that are used to procedurally generate game worlds.
2. To look at both 3D and 2D map generation and to decide on which model will be implemented in the end product of the project.
3. To assess the tool that can be used to generate game worlds such as for 3D worlds a game engine could be used to ease the process, or for a 2D world one would need to investigate the many languages that can be used to render and create a game world.
4. To look at the implementation of procedural generation in popular games such as Minecraft or Terraria, which implement procedural generation to a high level in their game worlds.

My initial approach to the project:

The first major question I have to answer is whether or not I am going to design a tool that generates a 2D or a 3D map, to do this I will have to investigate the many different tools available to both mediums and to decide on which one better suits my end goals for this project. Once I have then decided on which medium I am going to pursue I will then delve deeper into the tool available to that medium such as game engines, render libraries, relevant algorithms etc. The main goal of the first month of my project is to decide on which medium to use. Below will be some examples of tools that can be used in the generation of 2D and 3D game worlds and a brief description of them, more details will be supplied in the actual report stage of the project. This section will also include some examples of some of the tools in practice.

Game engines and 3D worlds:

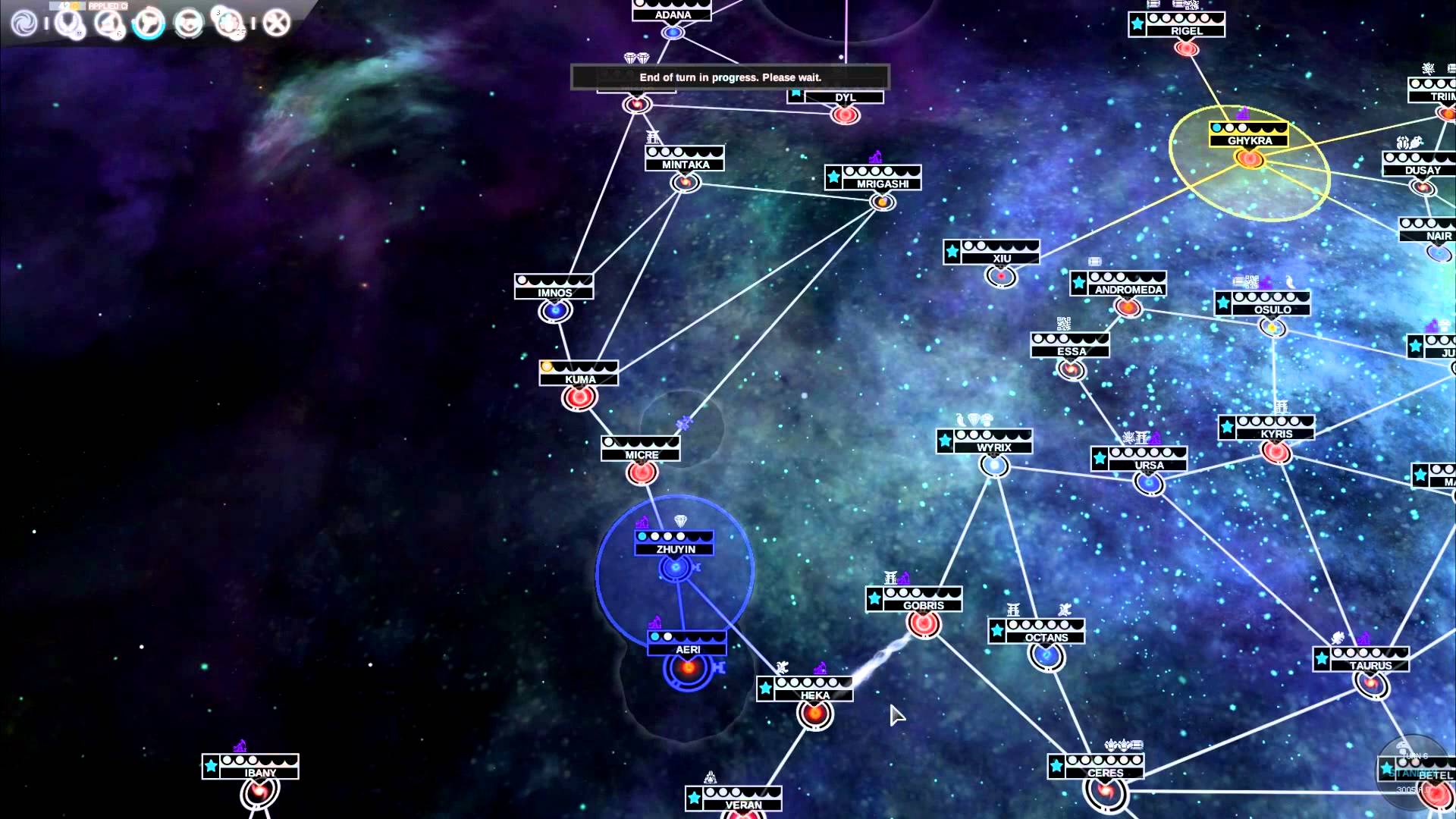
I initially looked at a 3D game world and looked into various game engines which I will go into more detail on below. I also looked into the Perlin noise algorithm and found that a lot of developers had stopped using Perlin noise to generate worlds and have moved on to its successor simplex noise. I found a C# implementation of simplex noise and studied it to figure out how it worked.

Godot engine;

This was the first game engine that I investigated. It is an open source game engine with and MIT license. It uses C++ or its own scripting language GDscript to create games, the more I looked into this engine I found that very little games were created in the engine itself, I acquired a copy of the engine off of steam and had a look around it features, the overall layout of the engine was quite intuitive and had a lot of the feature easily accessible. The website (<https://godotengine.org/>) had a few tutorials available but not an extensive amount of help. I looked into procedural generation for Godot but didn’t come across anything substantial and so decided that the game engine would not suit my needs and moved on.

Unity engine:

Unity is a cross platform game engine designed and developed by unity technologies. It has four different “plans” a personal plan which is free and has all engine feature, a plus plan which is $35 a month which adds features such as performance reporting, a pro plan which is $125 per month and has all the features of personal and plus which additional features such as source code access and premium support, the final plan is the enterprise plan which is the same as premium but is used for business and studios. Unity uses C# to code its games and can also use its own scripting language UnityScript. When looking into unity I obtained the personal plan and investigated the user interface which was very well designed and highlighted key features very well. There were also extensive tutorials available online and on the unity website which greatly helped with my understanding of the game engine and its inner workings, unfortunately I was not able to gain access to the source code for unity without having to pay for the premium plan. Many modern games are created using unity such as endless space which uses a type of node based procedural generation to create a “graph” of connected stars that the player can visit, and then based on the type of star present the game would produce a set of relevant planets to accompany the star. Although not the type of generation that I am looking at it was still interesting to see the tools that were used to implement this system while investigating the unity engine.

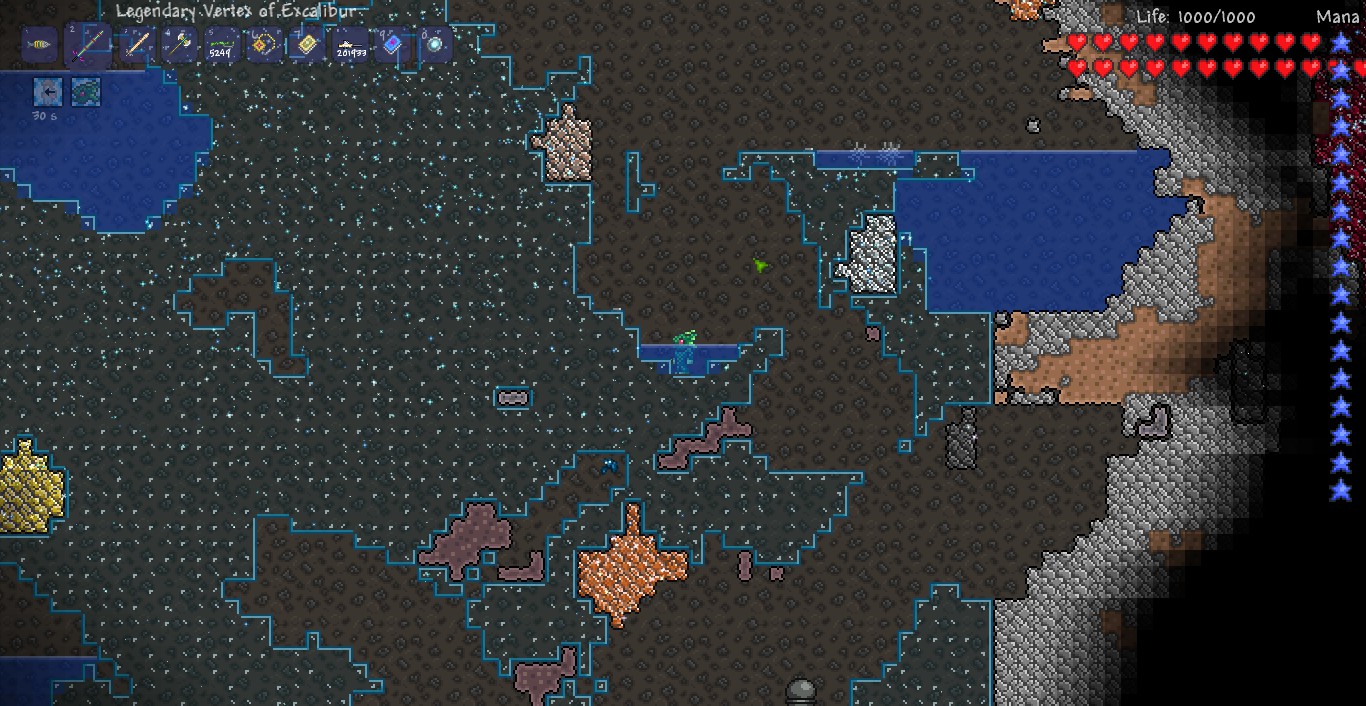


(Example of a map created using unity – Endless space)

Libraries and 2D game world:

Microsoft XNA game studio:

The Microsoft XNA game studio is an open source integrated development environment which extends visual studio with support for the XNA framework and tools. It is used to create games for windows, Xbox 360 and windows phone. It includes the XNA framework 4.0 which is a set of managed libraries designed for game development based on .NET 4 and is coded in C#. Many popular 2D games have been coded using this environment such as Terraria, bastion and Magika. Terraria produces a fully procedurally generated map using complex algorithms to produce a new unique map each time the player creates a world. The worlds host multiple complex structures such as cave systems and buildings that spawn under, above and at ground level, some of these structures are not triggered until a milestone is passed in the game which will then cause the structure to spawn. The downside of Microsoft XNA studio is that it is discontinued but it is still available for download, but any bugs that persist in the environment now will more than likely stay there.



(Basic cave found in Terraria generated with XNA)

PixiJS:

PixiJS is a JavaScript render library used for 2D rendering, I has many tools which include, multi-platform support, sprite sheet support, an auto detect renderer feature and many more. It has been used in the past for many HTML 5 games. Many websites have used it for its powerful render capabilities. To utilize PixiJS all you need is to simply download the PIXI.js file and use it as the script source in your web page. One slight problem with PixiJS is that it will not locally run on google chrome and it will only run locally without a webserver using the Firefox browser.

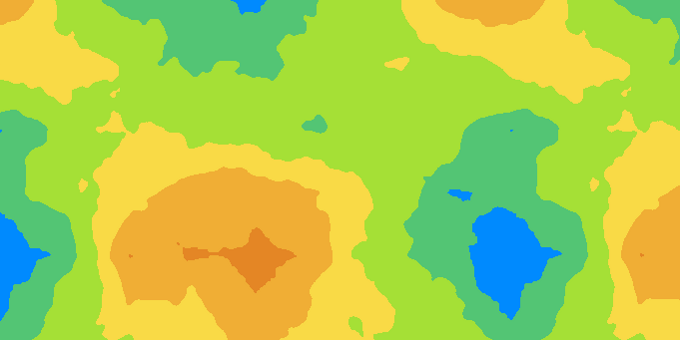


(tilemap created using PixiJS (<http://www.html5gamedevs.com/topic/8290-interactive-tiles/)>)

Algorithms used to procedurally generate worlds:

Perlin noise:

Perlin noise named after its developer Ken Perlin in 1983, is an algorithm that is used to produce natural looking textures on generated surfaces. It is used in motion picture special effects as well as in the area of procedural map generation in computer games. Perlin noise can be used for any number of dimensions but is most commonly used in 2,3 and 4 dimensions, the computation complexity for Perlin noise is O(2^n) for n dimensions.



(simple map generated by Perlin noise. (<http://devmag.org.za/2009/04/25/perlin-noise/)>)

Simplex noise:

Simplex noise is the successor to Perlin noise made by Ken Perlin in 2001 to address some of the issues that his previous noise algorithm had. The improvement in simplex noise include a lower computational complexity of O(n^2) and it carries to higher dimension with much less cost in computational power. It has no noticeable directional artifacts but noise in different dimensions are distinct, i.e. a cross section of 3D noise would have a different look that 2D noise. There exist a patent that covers the use of simplex noise in 3D and higher with respect to image synthesis. This gave rise to an adaptation of Simplex noise called OpenSimplex noise that is a way to use the core simplex noise ideas without coming into conflict with the patent.



(noise map generated using Simplex noise (<https://cmaher.github.io/posts/working-with-simplex-noise/)>)

Popular games that use procedural world generation:

Minecraft (non game engine procedural generation example):

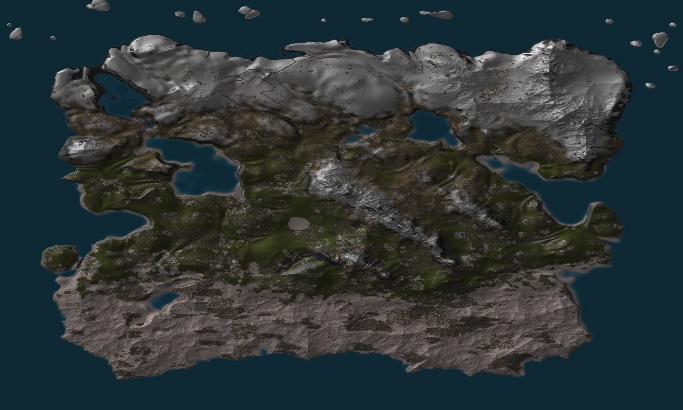
Minecraft is without a doubt the most popular procedurally generated game of the last few years and propelled the ideas of complex seed based map generation into the world media. The basic concept of Minecraft was so well implemented that it spawned countless games that imitated its core design values. Minecraft which was developed by Mark Persson (Notch) and coded completely in java and consisted of a procedurally generated world of textured cubes. Maps were based on a seed that was generated alongside the map itself and this seed could be shared among people and put into the map generator on any machine and produce an identical map. This is a key feature of the generation used in Minecraft and allowed players to share the complex and beautiful maps that were generated by the tool.



Above is an example of a chunk of a world that the Minecraft engine could generate, which included caves, mountains, rivers, oceans and many more complex features that are still being added to the generation tool to this day.

Rust (game engine procedural generation example):

Rust is a multiplayer only open world survival game in which the word can be procedurally generated, Rust was created using the unity engine. Rust is currently available on steam and is currently in early access. The procedural map generation in rust is still experimental and not all servers for the game have implemented it, the map generation is based on a random number generator or seed. There are also options that can be set by the user during the generation process.



Above is an example of a map that can be created using the map generator in rust. (http://rust.wikia.com/wiki/Procedural\_Map)

My project:

For my project I have decided to focus on the 2D element of procedural map generation and to create a tool to fulfill the specifications outlined in the project definition above. I will experiment with many of the 2D tools available to me and then create a program in whichever tool suits my needs the most. I will also investigate the methods of 3D map generation as a way to compare the two different processes but will go more in depth in the 2D section. The textures/graphics that I will be using will most likely be open source art. The program I create will use some abstract features such a weather patterns that influence grass or tree growth, or a system of shadows that imitate cloud coverage moving along the terrain etc. I have set out for myself the following rough timeline and deadlines that I aim to follow throughout my project.

Project Timeline:

October:

To fully investigate and understand the requirements of the assignment and to investigate the tools available to me.

November:

To have decided on 2D or 3D and have chosen the tool that I will use to implement my physical part of project. And have investigated algorithms and techniques used in the area of procedural map generation and start a log of the basic properties of these.

December:

Have a running prototype of the system that I am going to create with basic functionality. Do background reading during the break. Investigate other procedurally generated maps in popular games and make short-hand notes.

January:

Implement noise algorithms and other techniques into the prototype created in December and to have some basic textures worked into the map (open source or otherwise)

February:

Start transcribing short hand notes into useable format for the final report and to build on complexity of program, and to have a first draft of final report.

March:

Compile a second draft of final report addressing issues highlighted. Look through code of physical project and make adjustments and/or improvements.

April:

Final draft of report due to be handed in, look over code, clean it up and leave ready for submission. Prepare the short video presentation for the video.