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 algorithms 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 algorithms that people use 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 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 and to the get some sort of model up and running to help visualize the problems that may be encountered.

3D procedurally generated game world:

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 studies 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 licensed by mit. 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. The was also extensive tutorials available online and in the unity website which greatly helped with ym 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 he 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.

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 include 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 generates fully procedurally generated maps on 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.