Cubulus: Procedural World Generation Using Marching Cubes

Keeton Purvis – Computer Science
Advised by Prof Heather Guarnera

Abstract

Procedural generation is a popular way of algorithmically generating content such as worlds, textures, and sounds. This method allows the developers to save time and computer storage as content no longer requires manual creation or storage. It also enhances replay-ability in games by incorporating an aspect of randomization to content creation. This work investigates good game design and two common methods used to procedurally generate a world: Perlin Noise and Marching Cubes. Perlin Noise is an algorithm to generate data which appears natural. Marching Cubes is a method of terrain data storage so that the game can efficiently render the terrain. The Unity game engine can be used to demonstrate how these two methods are able to help each other create a procedurally generated world.

Procedural Generation

Figure 1. Generated World in Minecraft

Procedural generation is a way of using algorithms and equations to create some item. It is often used to create virtual terrain. It also can be used for images, music and other things. Think of the game Minecraft where the landscape seems to go on forever and never look the same-that is procedurally generated terrain. Procedural generation has multiple useful features, one is that it saves developers time, conventionally they have to do everything by hand, however procedural generation can either do all the work, or can create a general layout that can then be edited which saves a lot of time. Procedural generation can also save a lot of storage space when creating a game, this is because rather than storing the object directly, the method of creating it is stored. A game called .kkrieger (Figure 2.) created in 2004 for a game competition was rendered entirely through procedural generation, the textures, world, and music. It only took up 97,280 bytes of space, and the developers said if done conventionally it would have been 200-300 [1] MB, which means it takes up more than 9/10 less space.

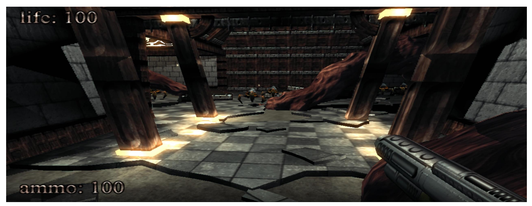
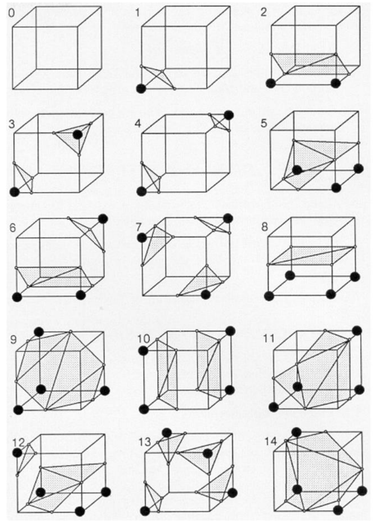


Figure 2. Image from the game .kkrieger.

Perlin Noise and Marching Cubes

This project used two methods to create a procedurally generated world, Perlin Noise and Marching Cubes. Perlin Noise version of visual noise such as the static that can be seen on older Televisions when there is no channel. Perlin Noise was made by Ken Perlin in 1983 [2] when he was working on the movie Tron, he needed better textures for the world, so he created Perlin Noise. The difference between this and normal static noise, is that it looks more natural, this is useful for procedural generation because it can also be used to make a natural looking landscape. Marching Cubes was originally made to model patients during medical scans but can also be used to generate terrain along with Perlin Noise. With Marching Cubes the world is separated into cubes, they are then rendered by getting data and setting the 8 points of the cube as 'underground' or 'above ground'. Terrain triangles will then be drawn to separate the above ground from the below ground.

Right: **Figure 3.** All cases of Marching Cubes



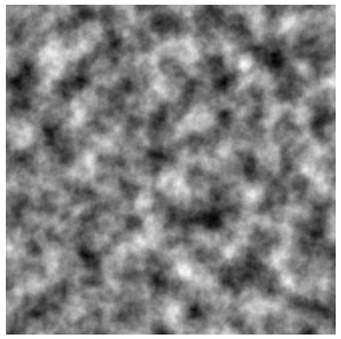


Figure 4. Example of generated Perlin Noise

Game Design

While developing the game I also tried to learn as much as possible about game design in general, to do so I read through two books about game design from Jesse Schell [4] who was the lead designer of Toontown Online, and Scott Rogers [5] who worked on God of War, Mrs Pac man and many other games.

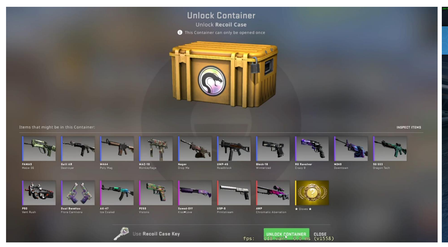


Figure 5. Example of loot boxes from Counter Strike: Global Offensive

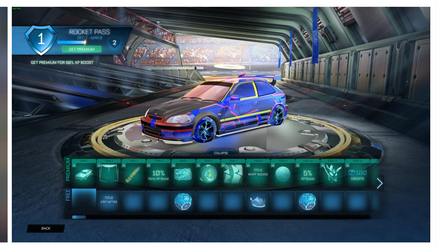


Figure 6. Example of a 'battle pass' from Rocket League

The most important sections of the books related to the game I was developing these sections being; what is game design, first person vs third person, purpose of different controls, and monetization. Monetization being the most intriguing as modern games have many ways to do so such as loot boxes in CS:GO (Figure 5) and battle passes such as in Rocket League (Figure 6).
I also reviewed a good video from Torulf Jernström on modern monetization in games.

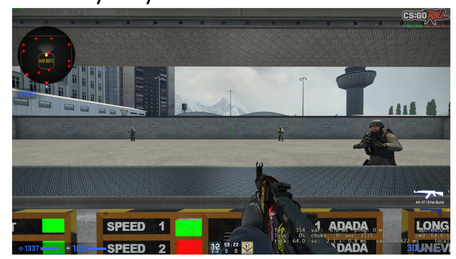


Figure 7. First Person in CS:GO

Results and future work

Below are images of a map created in the Unity Game Engine (uses C#) and is made using Perlin Noise and Marching Cubes, along with some other features.

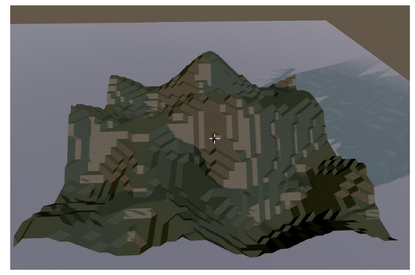
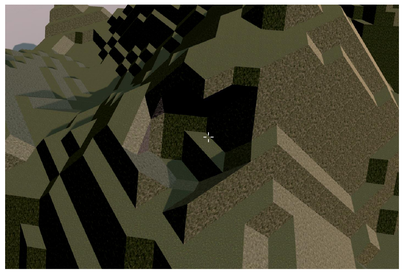


Figure 7. Image of the generated map

Figure 7. Image of the generated map with edits



Possible Future Work.
With this project there is a great deal of possible future additions. The main ones would be; infinite world/chunks, saving, water, and possibly spherical worlds such as in the game Astroneer in Figure 7.

Features of game:

- Terrain itself.
- Day/Night Cycle.
- Texture based on angle of terrain.
- Moving camera and character.
- Editable terrain.
- Placeable object.

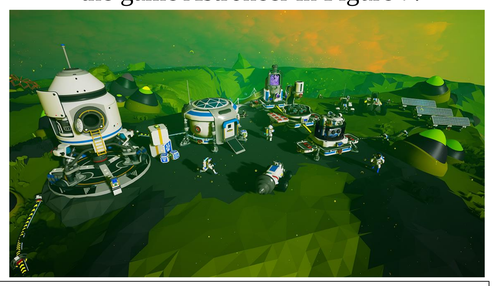


Figure 7. Image of the game Astroneer [Steam]

References
[1] *Nostalgia Nerd. kkriger: Makin an Impossible FPS | Nostalgia Nerd.* https://www.youtube.com/watch?v=bD1wWY1YD-M
[2] Ken Perlin. "Improving Noise". In: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '02. San Antonio, Texas: Association for Computing Machinery, 2002, pp. 681–682. isbn: 1581135211. doi: 10.1145/566570.566636. url: https://doi-org.wooster.idm.oclc.org/10.1145/566570.566636
[3] William E. Lorensen and Harvey E. Cline. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". In: SIGGRAPH Comput. Graph. 21.4 (1987), pp. 163–169. issn: 0097-8930. doi:10.1145/37402.37422. url:https://doi.org/10.1145/37402.37422
[4] Jesse Schell. The Art of Game Design-A Book of Lenses. Boca Raton, FL: CRC Press, 2019
[5] Scott Rogers. Level Up! The guide to great video game design. West Sussex, United Kingdom: John Wiley and Sons, 2014
[6] Torulf Jernström. Let's go whaling: Tricks for monetising mobile game players with free-to-play. YouTube. 2016. url: https://www.youtube.com/watch?v=xNjl03CGkb4