**Procedural generation of an alien planet**

By Haitham Al Saidi

# Motivation and rationale

## Context

Procedural generation is an algorithmic method of generating content using a mixture of human generated assets and computer pseudo-randomness. One of its most prevalent applications are in that of generating the levels or worlds, amongst other aspects, of videogames. Its prevalence is due to it being an accessible technology that enables game developers to generate large amounts of re-playable content with lots of variety, and all on a budget. This has enabled indie studios to generate game worlds on scales that would not have been possible on their budgets. A great example of this is Minecraft by Mojang, which started out as an unemployed developer’s passion project, to harbouring one of the biggest gaming communities on the planet and being bought by Microsoft for US$2.5 billion in 2014. [1][2] I believe it is vital that smaller developers, who are largely the source of the gaming industries new ideas, have tools like procedural generation available to them in order to compete with larger game companies with higher budgets. [3]

## The problem

Procedural generation has significant drawbacks a developer would need to consider. A poor implementation of procedural generation could result in a repetitive game with a large yet empty world of no substance. Considerable time and resources need to go into getting it right to prevent this. [4] Furthermore, due to its random nature, it leaves the developer with less control over quality and choice. For example, it would be difficult to create a game with scripted events in a randomly generated world, as there is no way to guarantee the order in which they would come across them, or if they come across them at all. This means you are limited in the types of gameplay you can offer with a procedurally generated game. Finally, procedurally generated games tend to be very taxing on hardware. This is because instead of simply loading premade assets, the game is constantly generating new content, which can have a very large performance impact. This would likely explain why procedurally generated games tend not to have stunning graphics. These are amongst some of the issues I will need to overcome when creating my game.

## Rationale

I intend to create a game which consists of a procedurally generated alien planet, tackling some of the prevalent issues of procedural generation mentioned above, while demonstrating its power and effectiveness as a tool for creating a vast and varying game world with limited resources and essentially no budget. The game will have variations in landscape and different types of terrain, or biomes. I will create the game using the Unity game engine and the C# programming language.

# Aims and objectives

## Aim

## To create a video game with Unity that utilizes procedural generation with Perlin noise to achieve pseudo-random endless terrain generation.

## Objectives

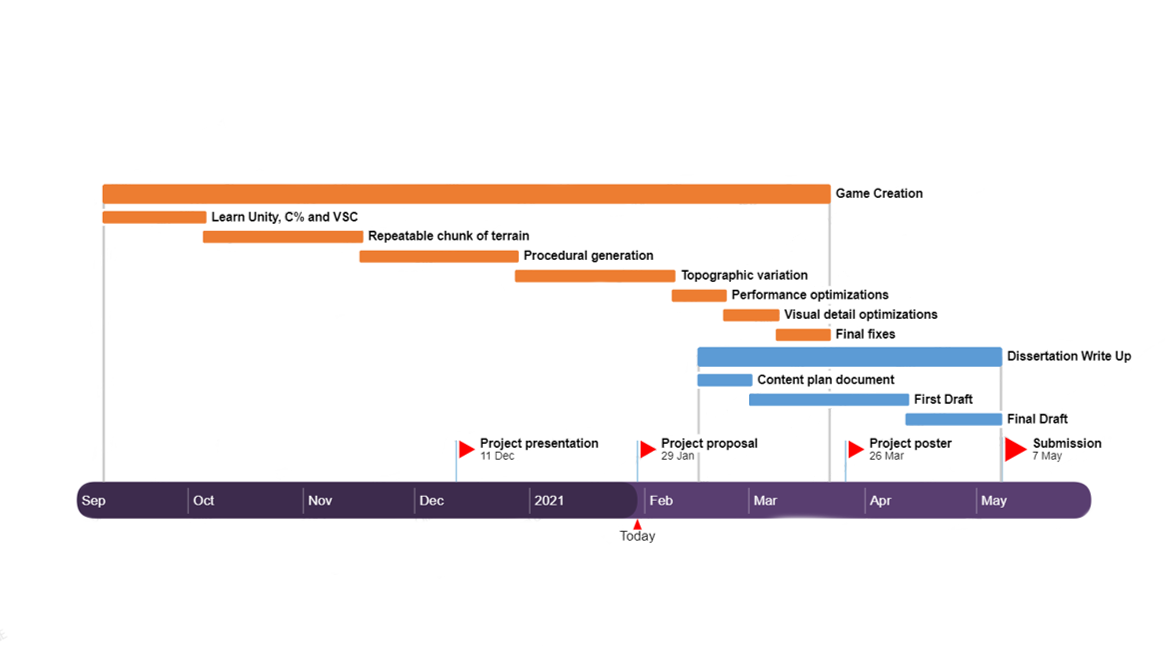
1. To accommodate myself with Unity, C# and Visual Studio to use these tools efficiently and effectively.
2. To create a chunk of terrain using a triangle mesh.
3. To use Perlin noise to create variation in topography, including biomes and typical terrain features such as lakes, oceans, beaches, plains, hills and mountains.
4. To implement infinite, repeated and pseudo-randomly generated chunks of terrain.
5. To ensure an acceptable performance by utilizing techniques such as multithreading and variable levels of detail. The measurable goal is, with an Intel i7-4700k CPU and a NVIDIA GTX-970 (7–8-year-old hardware):
   1. An average FPS of 60.
   2. A 1% low FPS of no less than 50.
   3. A 0.1% low FPS of no less than 45.
   4. FPS results are averaged over 5 tests, each involving 5 minutes of constantly moving in one direction and generating new chunks.
6. To improve visual detail by implementing features such as colour and texture shaders, as well as flat-shading lighting. Two good examples of what I would deem outstanding visual detail for indie procedurally generated games that utilize flat-shading are ‘Polytopia’ and ‘For The King’. [5][6]

# Background

|  |  |
| --- | --- |
| **Source** | **Significance** |
| **Type:** YouTube channel  Brackeys [7] | **Summary:** The Brackeys YouTube channel is dedicated to game dev tutorials, including amazing free courses such as ‘How to program in C#’.  **Relevancy:** The Brackeys channel was my main source when learning how to program in C# and was my secondary source for familiarizing myself with Unity after the documentation. I learned the concept of meshes and how to create them in Unity from Brackeys, as well as how to format C# code, tips and tricks with the Unity editor and other small issues whenever I ran into them. Brackeys is definitely a significant reason as to why I am interested in game engineering and as to why I am performing to well at this point. |
| **Type:** YouTube playlist  Procedural Terrain Generation by Sebastian Lague [8] | **Summary:** A playlist that takes a step-by-step detailed dive into one implementation of procedural generation.  **Relevancy:** This is a course that walks you through creating a procedurally generated world. While it has a very simple world as its outcome, it has been a very good source of insight into how procedural generation can be done. I have definitely used some ideas from this series in my own project. Sebastian has an amazing way of explaining things more than what is necessary to understand the current task at hand, giving me a deeper understanding of the Unity engine and some of the methods used in procedural generation. |
| **Type:** Paper  Procedural Generation of Dungeons by Roland van der Linden (2010) [9] | **Summary:** A paper that comes from the IEEE Transactions on Computational Intelligence and AI in Games journal with some insight on how procedurally generated dungeons are made.  **Relevancy:** While the nature of procedural generation used to create levels and worlds differ, the paper has been very useful to me in identifying ways you can generate randomness for any type of procedurally generated content. It is also very informative in how to control this randomness and modify it to different needs. This has proven especially useful to me while trying to create topographic variations in my game world. |
| **Type:** Book  Procedural Generation in Game Design by Tanya Short, Tarn Adams (2017) [10] | **Summary:** A book I purchased as a teenager while I was trying to get into game development. This book contains information on procedural generation of all types, from when to use it to how to use it.  **Relevancy:** My general understanding of procedural generation came from this book when I purchased it years ago while I wanted to get better at making Minecraft mods. This book is my best source of logic of use when creating procedural generation thanks to its clearly illustrated and explained pseudocode. This book also contains a lot of other interesting applications of procedural generation such as textures and audio, giving me a greater perspective of procedural generation as a whole. |

# Work plan

## Timeline



## Explanation

The orange bars indicate the objectives listed above for creating the game. The blue bars indicate my progress with compiling information and the actual writing of the dissertation. The red flags are submission deadlines.

The deadlines and start times are not absolute in the sense that I can begin and end processes earlier, but NOT later than the intended dates. I tend to do work in bursts and may get done with more work in certain weeks that I would others, which is the reason it is as such. As you can see, there is plenty of time for the initial creation of the game, including the procedural generation part. This is because I started my work very early on, as soon as I got my dissertation topic, and wanted to fully understand how my toolset worked.

At the moment, I am working on topographic variation and writing my content plan document. This is a document I am creating that details a more in-depth plan for my dissertation write-up now that I am more familiar with the content I have to write about. I am currently two weeks ahead of schedule. You may notice the larger amount of work coming in during the second and third semesters, and that is because I am only taking three modules then (as opposed to five in first semester). The time for performance optimizations is low as I will be considering them throughout the development stage, so that time is for refining the performance optimizations to achieve (or hopefully exceed) objective five’s requirements. Furthermore, in addition to this timeline, I have a Trello board with lists of sub-tasks for each objective that will help me keep track of what I have to do to achieve them.

I understand that my approach may be considered unorthodox, however it was been working for me since the second year of university. Before that I have tried many other methods of organizing my tasks with timelines to no avail. I call this the brute force method since I do not stop working until I have completed the project, even if I am ahead of schedule. This approach works for me because I can easily work for 14+ hours a day for a week without breaks (since I am enjoying this project), so long as I can then get the weekend to rest and reset.

At the moment, I am confident I will get my work done on time, and if not, ahead of schedule.

# References

[1] ewg.k12.ri.us. (2014). How Minecraft Started. [online] Available at: <https://sites.google.com/a/ewg.k12.ri.us/arc-minecraft/home/how-minecraft-started> [Accessed 29 January 2020]

[2] independent.co.uk (2014). Built to last: the Minecraft model. [online] Available at: https://www.independent.co.uk/news/business/analysis-and-features/built-last-minecraft-model-9788669.html [Accessed 29 January 2020]

[3] denofgeek.com (2014). Why indie videogaming is so important. [online] Available at: <https://www.denofgeek.com/games/why-indie-videogaming-is-so-important> [Accessed 29 January 2020]

[4] gamasutra.com (2016). The pros and cons of procedural generation in Overland. [online] Accessible at: <https://www.gamasutra.com/view/news/271814/The_pros_and_cons_of_procedural_generation_in_Overland.php> [Accessed 29 January 2020]

[5] polytipia.io (2020). Polytopia. [online] Available at: <https://polytopia.io/> [Accessed 29 January 2020]

[6] isonoakgames.com (2020). For The King. [online] Available at: <https://ironoakgames.com/> (Accessed 29 January)

[7] Brackeys (2020). Game Dev Tutorials and Free Assets. [online] Available at: <https://www.youtube.com/c/Brackeys/playlists> [Accessed 29 January 2020]

[8] Sebastian Lague (2019). Procedural Terrain Generation. [online] Available at: <https://www.youtube.com/playlist?list=PLFt_AvWsXl0eBW2EiBtl_sxmDtSgZBxB3> [Accessed 29 January 2020]

[9] Roland van der Linden, Ricardo Lopes, Rafael Bidarra (2013). Procedural Generation of Dungeons. [online] Available at: <https://ieeexplore.ieee.org/abstract/document/6661386> [Accessed 29 January 2020]

[10] Tanya Short, Tarn Adams (2017). Procedural Generation in Game Design. [online] Available at: <https://books.google.ae/books?hl=en&lr=&id=Rj4PEAAAQBAJ> [Accessed 29 January 2020]