# Abstract

This document is the proposal for the final project and dissertation for Games Software Engineering. Starting with an introduction to the procedural planet generation idea, and what it is expected the product will be, and how that compares to the industry currently. An initial literature review is then presented, to outline basic research done into the main ideas and concepts of procedural planet generation. From this research, a list of aims and objectives are detailed, describing all of the core components that will be featured in the final implementation. The document then finishes off with a brief methodology of how the project will be undertaken, as well as a project plan detailing the timetable and specific planning of the project.

Contents

[Abstract 1](#_Toc95402038)

[Introduction and Rationale 3](#_Toc95402039)

[Literature Review 3](#_Toc95402040)

[Sphere Generation 3](#_Toc95402041)

[Floating Point Precision 3](#_Toc95402042)

[Terrain Generation 4](#_Toc95402043)

[Atmospheric rendering 4](#_Toc95402044)

[Level of Detail 4](#_Toc95402045)

[Methodology 5](#_Toc95402046)

[Aims and Objectives 5](#_Toc95402047)

[Project Plan 7](#_Toc95402048)

# Introduction and Rationale

The topic for this final year project, is to conduct research into the most efficient and highest fidelity way of creating procedurally generated planets, that are also a realistic scale. The finished product should allow for the fast generation of scale detailed, randomly generated 3d planets. These planets will have features such as: multiple biomes, level of detail system, atmospheres and procedurally textured surfaces. This area is being investigated because, although all these areas have been individually researched and refined, they have yet to be combined into a single library/experience. Game such as Elite Dangerous feature to-scale planets but lack a lot of variation on the planet’s surface, whereas No Man’s Sky offers lots of variation but sacrifices its size. As a result, this project plans to combine the best of both worlds and produce something realistically sized and diverse.

# Literature Review

## Sphere Generation

In the field of procedural planet generation, there are many methods for the generation of the base sphere mesh. Some of the most popular ones are UV spheres, normalized cubes, spherified cubes and icosahedron (Cajaraville 2019). Unfortunately, all these different techniques have both positives and negatives associated with them. An ideal technique would be where the distribution of points is even, and as close to as possible to the points on the actual sphere. In addition to these techniques, an additional method using the Fibonacci sequence exists, called the Fibonacci sphere (S 2022). This is known as a “a well-known approach to generate a very uniform sampling of the sphere” (Keinert et al. 2015), and as such would be perfect for generating procedural terrain, if wrapped and triangulated correctly, which is unfortunately a very complex computational calculation(*Coding Adventure: Procedural Moons and Planets* 2020).

## Floating Point Precision

An issue that is bound to be encountered during the development process would be floating point precision errors. This is because floating point values would begin to lose the majority of their precision at 1000km (O’Neil 2022). Using some of the techniques outline by Sean O’Neil in his article, such as manipulating the view matrix. This would essentially mean moving the planet around the camera, instead of moving the camera around the planet (*Unite 2013 - Building a new universe in Kerbal Space Program* 2013). In addition to this main fix, scaling the planets and using doubles, should allow for the removal of any floating point precision errors that may occur. (*Unite 2013 - Building a new universe in Kerbal Space Program* 2013; O’Neil 2022)

## Terrain Generation

The most standard method for generating terrain is to use some version of noise in order to change the elevation of vertices (Fischer et al. 2020). Of this noise there are 5 main types, which all have their respective pros and cons (Vitacion and Liu 2019). An example of terrain generation based soley of noise is the planets in Kerbal Space program (*Unite 2013 - Building a new universe in Kerbal Space Program* 2013).But as seen in Vitacion and Liu’s paper, although this noise does provide a good base, noise alone will not allow for a detailed multi environment terrain(Vitacion and Liu 2019). In contrast, the work done by Fischer et al. demonstrates how to use this noise as a base, and add climate simulation on top of this layer to define biomes and more detailed terrain features (Fischer et al. 2020). The only downside of this implementation, is that is designed to work on a flat piece of terrain and as such, would have to be adapted to work on a spherical world.

## Atmospheric rendering

There are several methods used in the industry for atmospheric rendering, a variety of them would be unsuitable for the application being created here. One of these insufficient methods would be volumetric ray marching, as this method only allows for the rendering of the atmosphere from outside of the planet (Elek 2009). Fortunately, a method described in papers both by Elek and Schafhitzel et al., will allow for the creation of a mostly scientifically accurate atmosphere with good performance, as the first step (computing the scattering integral) is precomputed before runtime operation(Schafhitzel et al. 2007; Elek 2009). Another compelling method discussed in a paper by Florian Michelic, discusses an implementation that would not only allow for atmosphere rendering through a similar method discussed above, but also integrate the rendering of clouds into the algorithm (Michelic 2019).

## Level of Detail

A level of detail system (LOD) is crucial to the ability to run the intended executable, as the scale of the planets would not allow for a full detail model to be displayed at once due to the shear number of triangles being rendered. As such being able to switch between different complexities of mesh or dynamically change the mesh’s complexity would be necessary. There are 3 main techniques that can be done to the simplify an arbitrary mesh. These are: coarsen the mesh outside of the view frustrum, screen space geometric tolerances (measuring surface deviation from the original model then refining/coarsening the model based off an error value), and surface orientation (coarsening geometry not in view)(Hoppe 1998). Additionally, this technique described by Hoppe, allows for the smooth transitioning between these levels of detail using geomorphs.

# Methodology

Upon starting this project, the first thing that will be conducted is a further literature review. This is completed to gather all valid equations and methods that could be used in the final implementation. Additionally, some initial research will be conducted into the areas of the stretch goals, so that if time for their implementation is available, it can be completed. For the implementation of all the specified features, a design, create and test pattern will be observed. The implementation for generating the sphere would be the first to undergo this process. Along with this initial mesh, the level of detail system will also be implemented, as it too is a required component for the rest of the features. The next thing to be added would be then the actual terrain generation algorithms, both the base terrain features and the more specific biomes. Then the final major step in completing the main project would be undertaken, the addition of the atmosphere and possibly clouds. Then, if time allows, the stretch features will be completed in the order of multithreaded optimisations, Oceans and then Terrain manipulation.

# Aims and Objectives

The main goal of this project is to create an executable that demonstrates complete procedural planet generation system, that is both realistic in features and in scale. The final executable will allow the user to set parameters for the generation of the planet (such as size, colours, seed etc.) then proceed to explore the generated planet with a flying camera.

Following are all the components and problems that must be solved and researched in order to complete the project. These are the components completely necessary to the project and must be completed before the project deadline.

* Generating initial sphere mesh
  + Creating the vertices and geometry for the planet.
  + Fixing any floating-point precision errors that would appear due to the scale of the planet.
  + Setup a level of detail (LOD) system to simplify and coarsen the mesh of the sphere, will allow for LOD on the finished terrain.
* Generated Terrain on Sphere
  + Basic Terrain Generation
    - The basic started terrain shape that would be spread across the entire body of the planet.
  + Biome generation
    - Create the biomes on the planets, and then tweak terrain details to refine the individual biomes
* Planet atmosphere system
  + Creating the sky colour and visible atmosphere, and visualisation of the sun.

In addition to these main components, there are some stretch goals that would ideally be featured in the final version but will only be added if time allows. These additional components are listed below.

* Multithreaded generation
  + Using multiple cores in order to speed up the generation/rendering pipeline of the project.
* Terrain manipulation
  + Giving the user the ability to manipulate the generated terrain, by raising and lowering the vertices, and adding additional geometry
* Ocean support
  + Establishes an ocean level on the planet, and as a result adds large bodies of water between the landmasses/biomes

# Project Plan

Graphical user interface, application

Description automatically generatedShown below, and attached with this document (Gannt.png, Gannt.mpp), is a Gannt chart outlining the stages in which the project shall be completed, as well as all additional and associated deadlines. This Gannt chart shall be followed step by step to complete the project.

## Works Cited

Cajaraville, O. S., 2019. Four Ways to Create a Mesh for a Sphere. *Medium* [online]. Available from: https://medium.com/@oscarsc/four-ways-to-create-a-mesh-for-a-sphere-d7956b825db4 [Accessed 2 Feb 2022].

*Coding Adventure: Procedural Moons and Planets*, 2020. Available from: https://www.youtube.com/watch?v=lctXaT9pxA0 [Accessed 10 Feb 2022].

Elek, O., 2009. Rendering Parametrizable Planetary Atmospheres with Multiple Scattering in Real-Time, 8.

Fischer, R., Dittmann, P., Weller, R. and Zachmann, G., 2020. AutoBiomes: procedural generation of multi-biome landscapes. *The Visual Computer*, 36 (10–12), 2263–2272.

Hoppe, H., 1998. Smooth view-dependent level-of-detail control and its application to terrain rendering. *In*: *Proceedings Visualization ’98 (Cat. No.98CB36276)*. Presented at the Proceedings Visualization ’98 (Cat. No.98CB36276), 35–42.

Keinert, B., Innmann, M., Sänger, M. and Stamminger, M., 2015. Spherical fibonacci mapping. *ACM Transactions on Graphics*, 34 (6), 193:1-193:7.

Michelic, F., 2019. Real-Time Rendering of Procedurally Generated Planets, 8.

O’Neil, 2022. *A Real-Time Procedural Universe, Part Three: Matters of Scale* [online]. Available from: https://www.gamasutra.com/view/feature/131393/a\_realtime\_procedural\_universe\_.php [Accessed 4 Feb 2022].

S, A., 2022. *Delaunay+Voronoi on a sphere* [online]. Available from: https://www.redblobgames.com/x/1842-delaunay-voronoi-sphere/ [Accessed 4 Feb 2022].

Schafhitzel, T., Falk, M. and Ertl, T., 2007. Real-Time Rendering of Planets with Atmospheres, 8.

*Unite 2013 - Building a new universe in Kerbal Space Program*, 2013. Available from: https://www.youtube.com/watch?v=mXTxQko-JH0 [Accessed 10 Feb 2022].

Vitacion, R. J. and Liu, L., 2019. Procedural Generation of 3D Planetary-Scale Terrains. *In*: *2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*. Presented at the 2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT), 70–77.