# Abstract

Contents

[Abstract 1](#_Toc98239179)

[1.0 Introduction 3](#_Toc98239180)

[2.0 Literature Review 3](#_Toc98239181)

[2.1 Mesh Generation and Rendering 3](#_Toc98239182)

[2.2 Mesh Generation and Rendering 4](#_Toc98239183)

[2.3 Procedural generation techniques 4](#_Toc98239184)

[2.4 Atmospheric Rendering 4](#_Toc98239185)

[3.0 Design and Implementation 4](#_Toc98239186)

[4.0 Testing 4](#_Toc98239187)

[5.0 Evaluation 4](#_Toc98239188)

[6.0 Conclusion 4](#_Toc98239189)

# 1.0 Introduction

# 2.0 Literature Review

## 2.1 Mesh Generation and Rendering

The most basic component of the system that will be implemented as part of this dissertation is the creation and generation of a sphere mesh. This is crucial as all planets are typically a spherical shape, due to gravitational forces pulling material to the centre of the planet (Sears 2022). For this element of the project, there exists a wide variety of techniques and algorithms to make this initial sphere. Such techniques include: UV spheres, normalized cubes, spherified cubes and icosahedron (Cajaraville 2019). These algorithms can have their effectiveness evaluated based on their: computational efficiency, distribution of vertices, and how close the generated vertices are to the unit sphere. A benefit of the both the cube algorithms, is the ease to implement a Quadtree, which can be used as a level of detail system for changing the mesh’s complexity (Schneider 2006). One additional method is the Fibonacci sphere (Patel 2022). This algorithm allows for more evenly distributed vertices compared to the previously described methods and, as remarked by Keinert et al, is a “well-known approach to generate a very uniform sampling of the sphere” (2015, 7). Unfortunately, due to the non-linear generation of the vertices, triangulating these points would prove computationally difficult (Lague 2020). Another downside of this approach would be the difficulty for implementing a level of detail systems, caused directly by the generation method of the vertices. One promising technique is called the marching cubes algorithm. The method uses voxels, which is defined as “a value on a regular grid in three-dimensional space”(Anon. 2019). The algorithm works using a set of 8 voxels to form a cube, then generating a triangle based off these 8 values (Sin and Ng 2018). This technique is typically used on flat terrain, however the paper written by Sin and Ng demonstrate a method to transform the voxels into the unit sphere, allowing for the creation of spherical objects (2018). Unfortunately, the algorithm is known to be significantly slower than the other techniques described, due to the original algorithm having to traverse all the data to generate the mesh (Newman and Yi 2006). Although efforts have been made to speed up and improve this algorithm, a more traditional approach would work best for something of the scale intended for this project.

## 2.2 Floating Point Errors

Floating point errors may also occur during the generation and runtime of the planet generation. In the context of this project, due to floats only having 6 digits of accuracy, once you go further than 1000 kilometres out, you start to lose that accuracy (O’Neil 2022). Symptoms of this inaccuracy can be seen in a talk at Unite 2013 concerning the game Kerbal Space Program. This talk demonstrates a “Jitter” that occurs, which is a vibrating of the game object, that worsens the further out they bring the test spaceship (Unity 2013). In order to amend theses issues, the Kerbal Space Program developers then describe a solution that moves the player camera and game objects into different game spaces, depending on the current scale that is being dealt with (Unity 2013).

## 2.3 Procedural generation techniques

In Computer graphics, there are many methods for procedural content generation. One of the more popular techniques within this field is the use of noise functions (Reference needed). Noise is defined as, “the random number generator of computer graphics” (Lagae et al. 2010). Of these noise functions (such as Perlin, simplex and anisotropic) each function has their own characteristics, such as coherency.

## 2.4 Atmospheric Rendering

# 3.0 Design and Implementation

# 4.0 Testing

# 5.0 Evaluation

# 6.0 Conclusion