**Application Design Document**

1. **Introduction:**

This document outlines the general design of the developed application. For ease of reading and understanding, various areas of the application will be individually described in a semi-chronological manner.

1. **Base application:**
2. Implementing direct integration
   1. Issues this poses
   2. Solutions to those issues
3. Base UI
   1. Mostly for debugging – analysing and modifying the simulation

**Performance improvement**

Because in direct integration each individual body must take all the other bodies’ forces (no matter how miniscule) into account, with a given number of bodies N, the minimum number of required individual calculations can be calculated as N × (N - 1).   
Using the Big O Notation (Bae, S. 2019), the computational complexity of directly integrating Newton’s formulae is O(𝑁^2). This result is generally considered to be rather inefficient and it is exactly this computational issue that this project aims to address and solve by implementing performance improving algorithms described in the following paragraphs

1. **Tree Codes**
   1. **Definition**
2. As first described by its creators J. Barnes and P. Hut in 1986 (Nature 324, pp. 446– 449), “This technique uses a tree structured hierarchical subdivision of space into cubic cells, each of which is recursively divided into eight sub cells whenever more than one particle is found to occupy the same cell. This tree is constructed anew at every time step, avoiding ambiguity and tangling.“
3. Following this division, when calculating the gravitational forces acting upon a given body, faraway bodies are treated as one force defined by their combined mass and averaged position, whereas nearby objects are calculated directly.

Diagram

Description automatically generated

Figure 1: UML roughly outlining the order in which functions will be called on each frame.  
In this case, it is important to finish all gravitational dynamics calculations before applying the delta-position. Were the bodies moved before or during the dynamics calculation function, resulting changes in velocity would be inaccurate.

* 1. **Visual Representation**

One inherent strength of using Oct-tree codes is that it allows for an intuitive visualisation of its workings by simply displaying the Oct-tree division of the world. This will be extremely useful for debugging and analysis, as well as provide an understandable, intuitive visual cues to the potential future users.

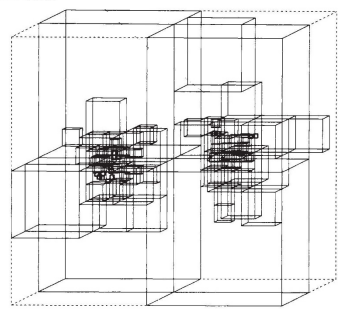


Figure 2: Graphical representation of the Oct-tree portioning within a 3-dimensional space. It clearly communicates where the highest density of bodies are located and how the space was divided. This figure was obtained from the original article (Nature 324, pp. 446– 449).

1. **Parallelisation:**
   1. Theoretical benefits
   2. Point of integration
2. **Evaluation:**
   1. Accuracy
      1. Frame Accuracy
      2. Longer Duration accuracy
         1. Run both in parallel or compare with pre-existing results?
   2. Performance
      1. Number of bodies simulated effectively
      2. Stability, FPS

**Gameplay:**

1. Recognising defining features and utilising them
   1. An important factor when designing gameplay is recognising mechanics, setting, strengths and the unique elements of the application.   
      In this case, the application is set in space.

The main mechanics are the realistic gravitational dynamics, as they are the main focus of this project and nearing its completion they will have taken up the largest portion of the development time. Furthermore, it is also safe to assume that these calculations will likely be a sizeable portion of the operations done on each frame.

Considering this, it becomes apparent that in order to develop satisfying gameplay, its elements should in some way utilize gravitational dynamics.

Despite the gameplay not having a clear aim yet, a few ideas of its implementation are present below:

1. Example 1: Slingshot Gods
   1. In this example of gameplay, the user will be playing as one of the Gods, being able to move around freely in Space, influence the passage of time, as well as interact with simulated bodies. The main mechanic would be shooting objects through space with the intent of the shot object hitting a desired goal (could be another God, a black hole or simply a large circular target. This would force the player to consider force of all nearby objects, utilizing their orbits.
2. Example 2: Drifting AI space station
   1. In this case, the user might play as a half-broken, self-sustaining space station that is operated purely by its AI autopilot. While this station is hurling through space, it attempts to collect chunks of space-debris (Broken satellites, etc) to upgrade itself. These upgrades could include a longer range for picking up objects, repairing the ship’s thrusters enabling movement, building a cannon to destroy asteroids (whether as means of self-defence or as a mining tool).

Adapting this direction, speeding up simulation time or various simulation analysis tools can be introduced in a story-friendly way, suggesting that it is the ship’s AI which decides to function on ‘Sleep’ mode when nothing of note is happening for a long period of time.

1. Example 3: A deeper dive into the simulation
   1. Finally, it is worth mentioning that the terms “Game” and “Gameplay” are relatively vague. As it is now, simulation games have a dedicated following of players, despite often having no gameplay, other than their sandbox nature.  
      It is therefore a valid possibility to follow this direction and develop more depth into the simulation. This could include more accurate collisions and breaking-apart of asteroids, having temperature/atmosphere on the larger planets, including different types of bodies (black holes, suns, moons), etc.

**References:**

1. Bae, S. (2019) ‘Big-O Notation’, in *JavaScript Data Structures and Algorithms*. Berkeley, CA: Apress, pp. 1–11. doi:10.1007/978-1-4842-3988-9\_1.
2. Barnes, J., Hut, P. (1986) ‘hierarchical O(N log N) force-calculation algorithm’. Nature 324, pp. 446– 449. https://doi.org/10.1038/324446a0