# Interim Report

## A Space Simulator Game for the Oculus Rift Virtual Reality Device

**Author: Alex Flight (77525)**
**Supervisor: Marco Gilardi**
**29/10/2013**

ABSTRACT
This project is undertaken with the intent to produce a fully interactive 'space simulator' game that the player will interact with using the mouse, keyboard and the Oculus Rift device [1], for which the game will specifically engineered and designed. The game will allow for the player to control a spacecraft in a space environment complete with various celestial bodies, semi-realistic controls allowing for 6 degrees of freedom in movement and various enemy targets that the player must seek out and destroy in order to progress within the game.

# TABLE OF CONTENTS

# TABLE OF FIGURES

# 1.0 INTRODUCTION

It is widely regarded that the Oculus Rift Virtual Reality (VR) device[1] represents a clear advance over previous attempts to produce a truly immersive 3D VR headset for the average consumer of video games [2]. Along with the introduction of the Oculus Rift a rekindled interest in Virtual Reality has emerged amongst those that see and have experienced the device's potential for bringing a high quality VR experience to consumers [3].

With access to an Oculus Development Kit, this project envisions to produce a game that takes full advantage of the features and capabilities of the Rift as the primary interface method. The Rift will replace a traditional computer monitor with fully immersive 3D via the headset device and provide a natural head tracking method of control over the player's view within the game world through use of the integrated motion sensors. This project will explore the potential offered by the Rift in a 'space simulator' game that puts the player inside a cockpit of a spacecraft inside a fully realised space environment with which to explore and seek out various enemy targets to destroy using a sci-fi arsenal including weapons such as lasers, rail guns and missiles.

Throughout this project the game itself will be described, designed and engineered into a final playable form that provides a player with an engrossing, immersive experience that seeks demonstrate the capabilities of the Oculus itself as a potential method of future video game interaction. To this end the capabilities of the Rift will be explored in order to provide an encompassing look at the challenges of engineering and designing a game specifically for the device, before looking at the potential future applications of the Oculus and other VR solutions in future games development.

Upon completion of the engineering of the game, this report will then explore the successes and failures of the project as a whole, focusing on critically analysing the various stages of the software engineering process, as well as the suitability of the final product to the intended goals. It will also analyse the effect that engineering a game for the Rift has on development as well as the considerations that have to be made in order to integrate the device into the development process at a fundamental level instead of considering it as an 'add-on' to an existing product.

[1] Within this report the Oculus Rift VR headset device will be referred to as simply 'the Oculus' or 'the Rift' for succinctness.

## 1.1 Genre and Gameplay Description

When considering a choice of game to produce for the Rift the device lends itself naturally to the player having a first person perspective on the game world. Traditional first person games can be divided into two broad categories: First Person Shooters (FPS), or those games which utilise the perspective of the player directly controlling an in-game body; and simulator games where again the player is typically located in the perspective of a head (disembodied or otherwise), yet this time affixed in the control seat of some vehicle which they occupy.

It is this type of simulator game that this project seeks to create, giving the player a seat inside the cockpit of a spacecraft and providing them with a feeling of presence within the game world. Within this environment the player is free to look around their surroundings freely from within their seat, with a feeling that they are truly inside a cockpit and directly controlling the actions of their spacecraft. Initial experiments with the Rift as well as the opinions concluded by others currently tend to favour the use of such a fixed perspective which remains consistent throughout gameplay. This offers the advantage of minimising the onset of motion sickness encountered by many when using a Rift in a traditional FPS-style game where the player may move 'unnaturally' [4].

Space simulator games typically entail the use of a player controlled spacecraft equipped with the ability to navigate around a space setting portraying various celestial objects such as asteroids, nebulae, planets and their moons. Typically level design in such a game is very non-linear and has a large area in which the player may explore freely in contrast to more rigidly designed levels in genres like the First-Person Shooter (FPS), Action/Adventure or some Role-Playing-Games (RPGs).

Within the described setting the player is typically equipped with a weapon system mounted to their spacecraft that allows them to engage various targets that include but are not limited to: enemy ships, asteroids, debris and various space stations or artificial satellites. The players' progress in destroying their targets is recorded in some manner until the overall objective of the scenario is met and the player advances to complete their 'mission' or objectives for the game. Excellent examples of such gameplay mechanics of the typical 'space combat simulator' are demonstrated in games such as the X-Wing/Tie Fighter series [5] and FreeSpace 1 & 2 [6] realeased in the 1990's/early 2000's, and more recently Strike Suit Zero in 2013 [7], although the genre has stagnated recently.

Similar to the 'space combat simulator' genre of games exists a sub-genre that gives the player the ability to freely choose what activities they want to pursue, sometimes labelled as a 'sandbox' or 'free-form' game. In such a game the player controls one or more ships that they may upgrade, purchase and sell and can also engage in trading, exploring and building an 'empire' of property and manufacturing, in addition to typical space combat. Examples of free-form space games include the classic Elite series [8], first produced in 1984 and more recently the X Series [9], famed for its relentless complexity.

In recent months the genre has seen a resurgence of interest [10] after a period of relative inactivity with many projects rebooting old franchises through crowd-funded efforts from their fan-base and renowned pioneers of the genre, such as Chris Roberts [11] with 'Star Citizen' [12] and David Braben with 'Elite: Dangerous' [13].

## 1.2 The Oculus Rift Virtual Reality Device

The Oculus Rift itself (Figure 1) is best described as a 'virtual reality 3D headset interface device', meaning that the device is worn on the head in order for the user to become full immersed inside a fully 3D virtual environment. At its very essence of functionality it is both an input and output device for a computer system, acting as both a full head orientation tracking device and a stereoscopic 3D output (Figure 2) for the images rendered by the computer to be projected to fill a large portion of the wearers field of view.

With these two elements combined the Oculus Rift is able to provide a fully realised implementation of a long sought-after virtual reality headset device that has been the goal of many since the very introduction of computer graphics over 50 years ago [14] [15]. Given Virtual Reality's comparatively long history, various implementations have been seen over the years from training devices [16] [17], to entertainment products [18]. With the introduction of the Oculus Rift expressly designed for video games, resurgent interest in the field of VR has re-emerged now that a device can live up to expectations long-held by many.


**FIGURE 1 - THE OCULUS RIFT DEVELOPMENT KIT [19]**

**FIGURE 2 – OUTPUT TO THE OCULUS RIFT IN THE 'TUSCANY' DEMO CAPTURED ON A NORMAL DISPLAY**

### 1.2.1 Sensors and Input

Whilst it is beyond the scope of this report to analyse the engineering and functions of the Oculus Rift at a detailed level, a simple description of the various parts of the device is necessary in order to understand the programming and implementation challenges and requirements imposed through the use of the Oculus Rift as an interface device. At the core of head tracking functionality of the Rift are a suite of three different sensors running at an incredibly high frequency of 1000Hz allowing for incredibly accurate interpretation of orientation in three dimensions into digital values. These sensors consist of a combination of solid state gyroscopes, accelerometers and magnetometers that act together to record movement, angular velocities, determine gravity and 'drift' of the headset to produce measurements of rotation and acceleration in the cardinal x, y and z axes of motion; commonly referred to as pitch, roll and yaw [20]. For more information in the technical aspects and challenges faced by the Oculus Rift, many articles are available on the Oculus VR company website [21] [22].

### 1.2.2 Display and Output

In recent years the popularity of portable electronics devices such as tablets and smartphones has led to a massive leap forward in the technologies behind small displays in terms of overall quality and, most importantly, value for money [23]. These displays offered an ideal candidate for the Oculus Rift as a head-mounted device, with the need for a compact screen in order to make the device light and wearable without discomfort. In the development kit used during this report the display within the Rift offers a 1280x800 resolution across a 7 inch screen, effectively offering a 640x800 (4:5 aspect ratio) per eye. Located in front of the display are two interchangeable lenses (allowing for simple dioptric correction in short-sighted users) that create a spherically-mapped image in front of the user's eyes, encompassing greater than 110° of diagonal Field-Of-View (FOV) (Figure 4) [24].

With this physical system in place the challenge for integrating the Oculus Rift into games is to produce an output that projects two discrete images (facilitating the Rift's stereoscopic 3D image) that are offset by a small amount and then warped using a 'barrel distortion' projection in order to compensate for the 'pincushion distortion' (Figure 3 - Barrel and Pincushion Distortion ) applied by the lenses in front of the display. When implemented successfully the image viewed through the Oculus is one of almost total immersion encompassing a large portion of an individual's field of view and hence leading to a feeling from the wearer that they are really 'inside' the game in question [26] [27].

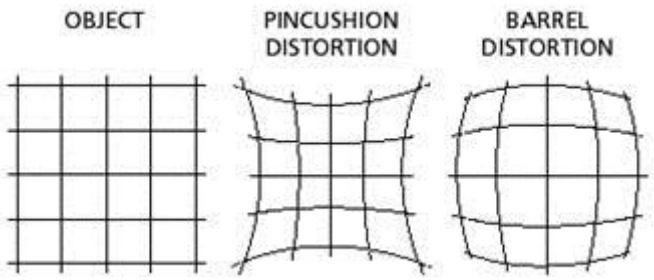


**FIGURE 3 - BARREL AND PINCUSHION DISTORTION [25]**

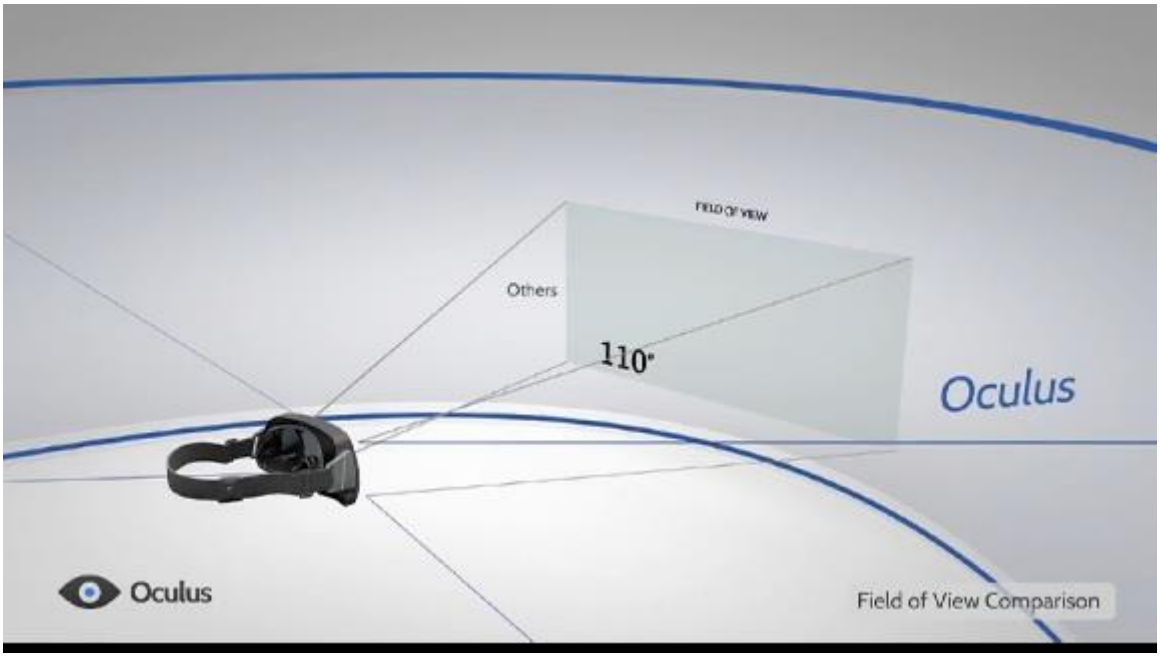


**FIGURE 4 - FIELD OF VIEW (FOV) OF THE OCULUS [28]**

## 1.3 Project Objectives

This project aims to produce a 3D space simulator game where the player is present within the cockpit of their own personal spacecraft and is tasked with navigating a convincing space environment in order to attack and destroy several enemy targets. Within this game the player will utilise the features of the Oculus Rift to be provided with a full stereoscopic 3D view of the game environment around the majority of their field of view with head tracking providing direct orientation of the player's camera, or head, within the game.

To this end the player will be represented within the game world as a pilot of a spacecraft located within a 3D cockpit complete with various displays detailing statistics relevant to their spacecraft and will have full control of their craft through the use of the mouse, keyboard and possibly a game controller, allowing them to move and steer in any direction they wish.

### 1.3.1 Implementation details

The various underlying components of the game will be programmed in the C++ programming language using the Microsoft Visual Studio 2012 IDE (using the VC110 compiler) and make extensive use of the open source *OGRE 3D* rendering engine to provide a high-level object-oriented interface to the computer's underlying graphics APIs [29]. On top of the *OGRE* engine a suitable physics engine such as *Bullet Physics* will be used to provide various physics-related features such as collision detection [30]. In addition various other libraries including *libnoise* and *Boost* will be utilised to provide coherent noise generation algorithms and various utility classes and features respectively [31] [32]. Alongside these libraries perhaps the most important library that will be used in this project is provided by the Oculus Software Development Kit (SDK) and *LibOVR* library used to integrate the Oculus Rift into the game [33].

In addition the following libraries will be considered and may be integrated into the project in order to facilitate a feature-complete game engine:

- *OpenAL Soft*- Audio library (http://www.openal-soft.org/)
- *FMOD* – Another alternative audio library (http://www.fmod.org/)
- *SFML* – Another alternative audio, input and multimedia library (http://sfml-dev.org/)
- *SDL* – Input library (http://www.libsdl.org/)
- *OIS* – Input library integrated into OGRE (http://sourceforge.net/projects/wgois/)
- *CEGUI* – GUI system (http://cegui.org.uk/wiki/Main_Page)
- *MYGUI* – Another GUI system for OGRE (http://www.ogre3d.org/tikiwiki/tiki-index.php?page=MyGUI)

These libraries and systems will be evaluated and considered during the development process and the complete game will likely include one audio library, one GUI library and one Input library that best fit the development goals and processes.

### 1.3.2 Primary Objectives

P 1. Implement a simple space shooter/simulator game using an appropriate game engine to allow for the player to control a spacecraft with independent head tracking provided by the Oculus Rift.

P 2. Include 'enemy' targets that the player must locate & navigate to then shoot and destroy in order to complete the game/level.

P 3. Provide a method of maintaining a 'score' or objectives for the player to complete through destruction of targets. Upon destruction of targets the player will be closer to completing their objectives.

P 4. Produce a space environment consisting of a star field skybox and other celestial bodies such as a nearby planet, asteroids, nebulae etc. in which gameplay will take place. Procedural generation of these assets will be considered where possible and appropriate.

P 5. Produce a working cockpit-based camera that takes direct input from the Oculus Rift motion tracking sensors to allow for orientation of the head to look around in every direction. A simple 3D model of the cockpit interior is needed for the basic functionality to make sense and immerse the player.

P 6. Integrate a solution that allows for output to an Oculus Rift display either through custom integration of the Oculus Rift API or an existing method within the chosen engine.

P 7. Implement/include a control scheme for the spacecraft based on a simple physics model of simplified flight-controls allowing for pitch, roll and yaw. In addition allow for translation along x, y and z axes which will allow for 6 degrees of freedom.

P 8. Produce a simple user-interface that is designed to be read from the various consoles and features of the cockpit itself much like a modern military fighter plane. This could include display monitors and a HUD like interface projected in front of the player. Traditional UI elements tend to be outside the OR's Field of View at the edge of the screen after barrel distortion has been applied.

P 9. Implement a weapons system that allows the player to shoot at various targets within the environment in order to destroy them to meet a specified objective (e.g. destroy x numbers of targets in y time)

### 1.3.3 EXTENSIONS

E 1. Implementation of basic enemy AI that allows for the player to dogfight with enemy ships in a simple manner. This could be expanded to increase difficulty and the sophistication of the AI as time allows.

E 2. Design of a fully immersive UI that integrates well with the Oculus Rift as a believable interface directly tied to the spacecraft itself.

E 3. Implementation of advanced graphics and lighting techniques to take advantage of modern graphics hardware where appropriate.

E 4. Expansion of control and physics to incorporate a fun representation of Newtonian flight through space (e.g. turning to face the enemy while conservation of momentum/inertia allows movement along current movement vector)

E 5. Investigate use of procedural generation of meshes and textures to generate realistic asteroids and planetary bodies.

E 6. Exploration of shaders and current graphics trends to enhance visual experience as well as explore what works well with the Oculus Rift.

E 7. Expansion of core gameplay to include scripted scenarios and missions as time allows.

E 8. Include varied environments in which to fly in.

E 9. Enhance the experience of the player in the cockpit with various effects such as g-forces pulling the head around, vibrations, creating a sense of acceleration and speed.

E 10. A missile system that allows the player to lock a target by looking at them from any direction within the cockpit making full use of the Oculus Rift as a targeting device while the player continues to track a target regardless of the spacecraft's facing.

E 11. A menu and interface implementation.

E 12. Adjustable difficulties.

E 13. Looking at tessellation to enhance detailed asteroid and terrain meshes produced on the GPU.

E 14. Gameplay & visual enhancements where appropriate to allow for a more immersive product.

E 15. Inclusion of more advanced shaders etc.

## 2.0 PROFESSIONAL CONSIDERATIONS

Throughout this project various considerations will be made in order to ensure that the project meets ethical standards and guidelines as defined by the Code of Conduct of the British Computing Society [34]. Such considerations include the decision to ensure the project itself is technically achievable and challenging in terms of a BSc final year project through the involvement of various technical and engineering challenges in order to complete the project. These can be seen during the software engineering approach to the analysis, design, and implementation and testing of the project as well as the use of new and current technologies in order to develop the systems in use by the game itself.

The ethical considerations of the development of the game in terms of game content are also applicable to the project as a whole. These considerations include that the theme, story, background and core gameplay are not in any way culturally insensitive or intended to cause offense or harm to an individual, group or society. In addition the use of the Oculus Rift device will also be considered during the development cycle and therefore attention will be paid to ensuring the user will not experience any adverse reactions or effects during gameplay. For example it is possible to cause the images projected to the Oculus Rift to be misaligned or offset to too great a degree and intentionally cause discomfort and nausea to the user; therefore steps must be taken in order to minimise or remove this effect completely during development [35].

Additionally considering the nature of the Oculus Rift as a development kit that has not undergone serious health and safety studies at this time, cautions should be provided to the user to ensure that they take regular rest periods when using the device and discontinue usage if they experience any adverse health effects or discomfort such as motion sickness, vertigo or dizziness.

# 3.0 Requirements Analysis

Before a large software project is undertaken a detailed analysis of the various requirements of the system must first be undertaken, with this project being no exception. Such requirements identified during this project can be classified into several categories: customer requirements, functional requirements and non-functional requirements.

## 3.1 Customer Requirements

In the case of games development the customer may be more helpfully labelled as a 'player' or 'user', yet they are still a customer for all intents and purposes. In terms of this project the customer was considered as a typical player of the game that would be interested in using the system purely for entertainment and immersion in 'another world':

- *The game should be enjoyable and fun* – The absolute goal of the project should be to produce a game that the player will enjoy playing.
- *A challenge should be presented to the player* – The game should be challenging in some aspects of the gameplay and should ensure that the player has a goal which is not trivial to reach.
- *The game should be easy to understand* – Controls, general gameplay and objectives should be straightforward.
- *Prompts and instructions should be provided to the player* – In order to instruct the player on the use of the game, appropriate instructions should be accessible to the player.
- *The player's health and wellbeing should be considered* – Particularly true in the case of the use of the head-mounted Oculus Rift, the game should not cause intentional harm or ill-effects to the player.

## 3.2 Non-Functional Requirements

Alongside the functional requirements that are necessary for the game to make sense and be playable, non-functional requirements must also be considered in order for the experience of the game to be successful and enjoyable:

- The game should run at a good frame rate
- The game should have fast load times
- The game should comply with modest system requirements in order to facilitate play across a moderate range of computer specifications
- Text and user interface elements should be readable and easily interpreted
- The system requires an Oculus Rift device in order to be playable
- The player should be using an appropriately supported operating system and architecture

## 3.3 FUNCTIONAL REQUIREMENTS

When undergoing testing the functional requirements are considered the goals which the system must reach in order to be considered successful:

- *A game world must be rendered in 3D representing a space scene in which the player experiences the game* – Implementation of a playable 3D space scene should be created within the game including necessary entities befitting to such an environment. This environment should then be the level upon which the gameplay itself is built. Such assets included in the game world must be created either through the importing of pre-built meshes and textures or utilising the CPU and GPU to generate the various game entities procedurally where possible. (Objectives *P 1, P 4, E 3, E 5, E 6, E 7, E 8, E 14*)
- *The player should have control of a spacecraft vehicle within the game world* – The player should be represented within the game world as a spacecraft entity that is directly controllable through the interface of their choosing. At a minimum basic keyboard controls should be provided to the player allowing them to maneuverer their craft and fire at enemy targets in order to destroy them. (Objectives *P 1, P 2, P 7, E 4*)
- *The player camera should be set up to accommodate the Oculus Rift VR interface device* –The main player camera in the world should be located within the cockpit of the player spacecraft and should offer the player the sense that they are surrounded by the environment using the unique features offered by the Rift. Display output to the Oculus Rift should accommodate the barrel distortion of two discrete images to the left and right eyes of the Oculus Rift using a 'side-by-side' stereoscopic rendering technique with appropriate offset for two eyes to create an effective 3D image. (Objectives *P 5, P 6, E 9, E 10, E 11*)
- *The player should be able to orient their camera through use of the head tracking capabilities of the Oculus Rift* – Orientation of the player camera within the cockpit should allow for the player to pitch, yaw and roll their whole head in order to look around their surroundings with as close to a 1:1 input response as possible using the input from the Rift. (Objectives *P 1, P 5, E 9, E 10*)
- *Basic 'space physics' should be implemented in the game* – Simplified Newtonian physics should be implemented into the game engine to allow for relatively realistic movement according to laws of motion and conservation of momentum where thrust, mass and acceleration are all taken into account in order to modify the relative velocities of the objects within the game. As such the player should be allowed 6 degrees of freedom in the movement of their spacecraft, allowing them to pitch, roll, yaw, accelerate, decelerate and translate their craft in the along the cardinal x, y and z axes. This should be expanded where time and features allow. (Objectives *P 1, P 7, E 4, E 9)*
- *A sci-fi weapon system (lasers/rail guns/missiles) should be controllable by the player in the game* – A weapon system suited to a sci-fi spacecraft should be represented in some way by the game and allow for damage and destruction of eligible player targets in the game. (Objectives *P 1, P 9, E 10*)
- *Enemy spacecraft, target drones and other entities should exist within the game world for the player to destroy* – Enemy entities such as other spacecraft or target drones should exist in the game world and be able to be destroyed through the interaction with the player's weapons systems. (Objectives *P 1, P 2, P 3, P 9, E 1, E 7, E 8, E 10, E 12*)

- *A UI display recording the score of the player should be implemented and notified to the player* – Upon destruction of enemy targets the player's 'score' should increase in some way either through allocation of points or the progression towards the completion of some defined goal of destroying n specific enemies before the player's mission is complete. (Objectives *P 3, P 8, P 9, E 2, E 10*)
- *Various visual and audible effects should be considered to enhance gameplay* – The inclusion of various graphical and audio effects and elements should be included in order to produce an effective and believable space combat simulator environment. Such effects could include post-processing to enhance the graphical fidelity of the scene, ship, weapons and engine sounds etc. (Objectives *P 1, P 4, E 3, E 5, E 6, E 7, E 8, E 9, E 11, E 13, E 14, E 15*)

## 3.4 UML DIAGRAMS

### 3.4.1 USE CASE MODELS

Use case models offer an overview of the system with regards to users and how they interact with it. In the case of this project a typical user is a 'Player' and they are interacting with the game in order to play it for personal entertainment.

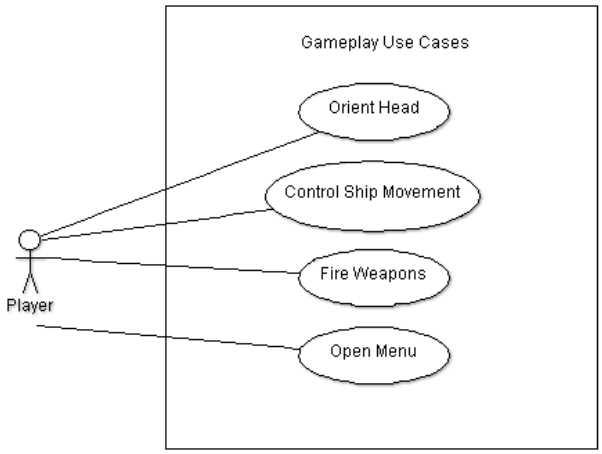


**FIGURE 5 - GENERAL GAMEPLAY USE CASE**

Figure 5 illustrates the 'general gameplay' use cases that a player will undertake within their interaction with the game. These are broadly characterised into the capabilities of the player to orient their head with the Oculus Rift, thus causing their perspective to change in the game; to use input devices to control their ship's movement and fire their weapons; and to open the game's menu system.

These core uses of the system will exist throughout gameplay and are accessible at any time whilst the game is in a 'gameplay' state.

**Error! Reference source not found.** illustrates the basic facilities the game executable and menus will provide within the game. At the most basic level this implies that the user may launch the game executable, start a new game from the main menu, or choose to exit the game executable entirely.

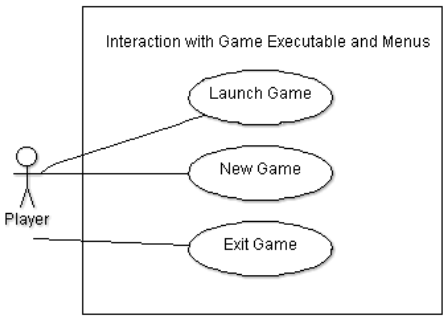


**FIGURE 6 - GAME MENUS AND EXECUTABLE**

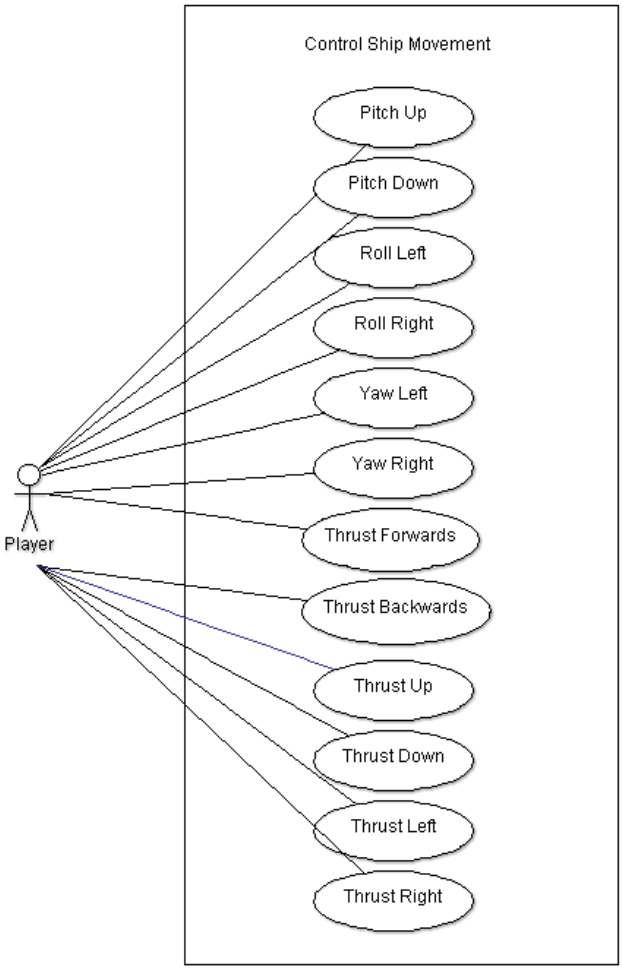


**FIGURE 7 - PLAYER CONTROL USE CASE**

Figure 7 expands upon the 'Control Ship Movement' use case within Figure 5 and represents the various controls that the player may use in order to cause their spacecraft to change positions within the game world.

The controls themselves will be accessible through keys presses on the keyboard or other interface device and are best described with the diagram in Figure 8.

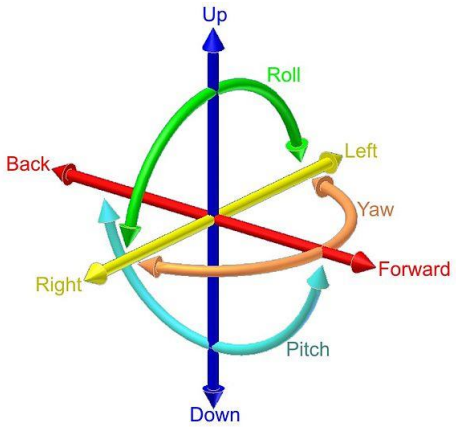


**FIGURE 8 - 6 DEGREES OF FREEDOM [36]**

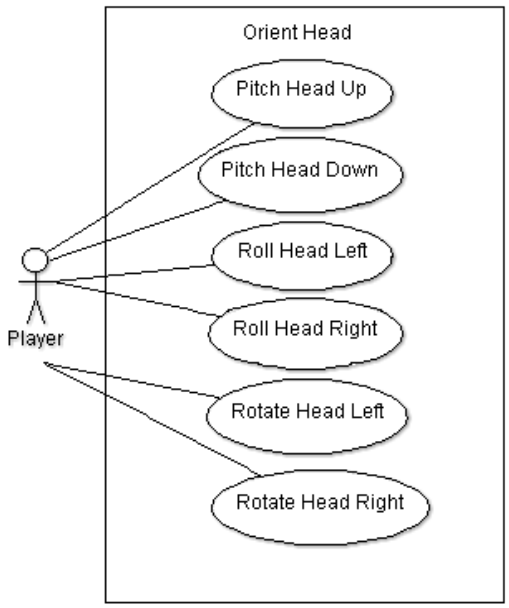


**FIGURE 9 - ORIENT HEAD/CAMERA USE CASE USING THE OCULUS RIFT**

Figure 9 represents the player's interaction with the game using the tracking features of the Rift; this includes the player moving their head from left to right (yaw), up and down (pitch) and side-to-side (roll). These actions will then be directly translated to equivalent movements of the player's head in the cockpit of their spacecraft utilising the Oculus libraries and camera integration.

### 3.4.2 State Diagram

Accompanying the use case diagrams is an overall state diagram which represents the various states the game is within when it is running from initialisation through to termination of the program.

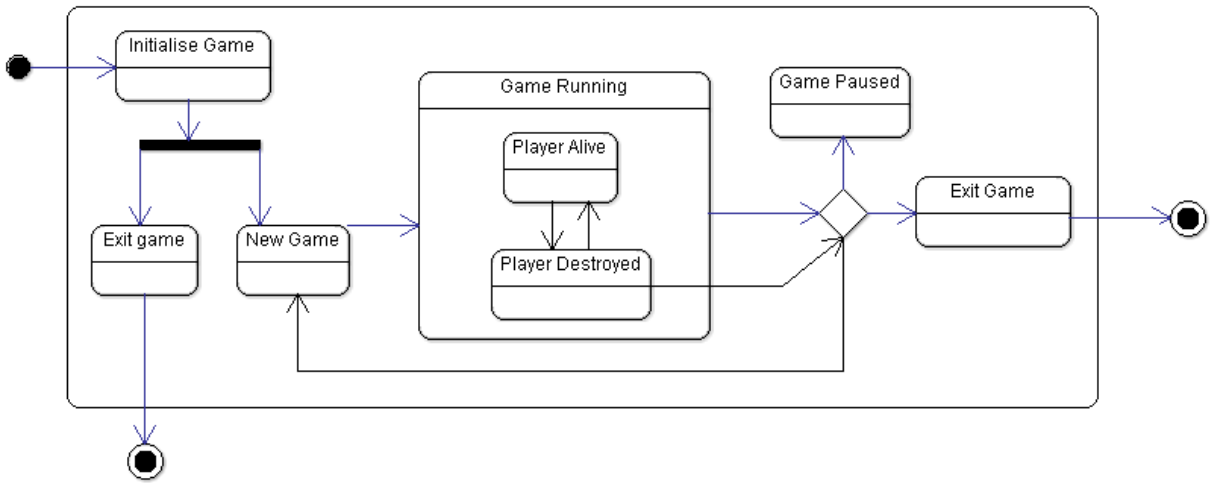


**FIGURE 10 - STATE DIAGRAM OF THE SPACE GAME**

Represented in Figure 10 are the various states in which the game exists. These include the main menu of the game which is the first state of the game upon execution, from which the player may begin a new game or exit the application entirely. From within the 'Game Running' state the player may be either dead or alive with the ability to transition to a 'Pause' state at their will or if they are destroyed. From this pause state the player may choose to begin a new game or exit the game entirely, terminating the program.

### 3.4.3 CLASS DIAGRAM

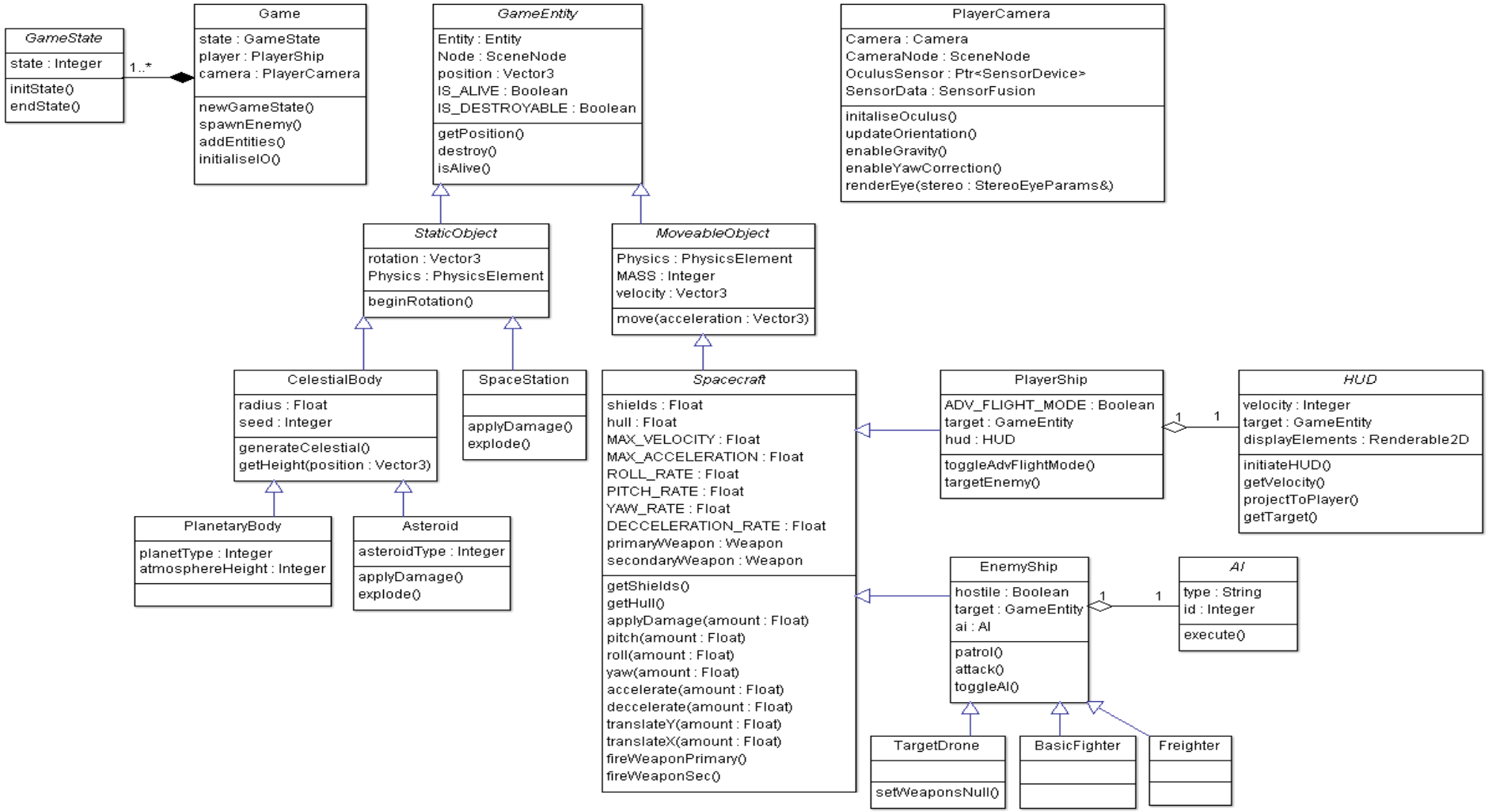


**FIGURE 11 - OVERALL CLASS DIAGRAM IDENTIFYING VARIOUS COMPONENTS OF THE GAME**

### 3.4.4 Class Diagram Continued

Figure 11 represents the initial class diagram developed during the initial design and analysis stages of the game's development. This diagram shows a very typical Object-Oriented approach to modelling the core features of the game into component systems and parent-child object hierarchies. Abstract classes are represented within the diagram in *italics*. As can be observed in the majority of the diagram, all game objects including the Player, Enemies, Planets and Asteroids inherit properties from a 'GameEntity' abstract class which in turn is inherited by 'Moveable' and 'Static' objects which in effect divide entities in the game world into those objects which move during the course of gameplay and those which do not.

The main 'Spacecraft' abstract class defines the properties and functions which all spacecraft objects in the game world inherit, and is common to both 'EnemyShip' classes and the 'PlayerShip'. However it is here that the differences in the implementations of the various spacecraft in the game are defined with the PlayerShip receiving input from the Player themselves and may include a HUD object through Aggregation (a 'has-a' relationship). The 'EnemyShip' instances may include a reference to an AI object (again through aggregation) that controls the behaviour of their ship. Additionally there exist various subclasses of the EnemyShip class that define different behaviours, meshes and capabilities.

Looking at the *StaticObject* abstract class it can be seen that this type divides into two discrete subclasses that include 'CelesitalBody' classes representing naturally occurring bodies, and 'SpaceStation' objects representing artificial objects that behave like static bodies. This separation allows for the implementation of potential procedural generation of the meshes and appearance of the 'CelestialBody' types.

Also represented within Figure 11 is the definition of the *GameState* abstract class which represents various states that the main 'Game' class, which represents and initialises the game itself, must contain (a composition relationship, meaning the Game *must* be in a *GameState* of some kind at all times).

The final class represented in the class diagram in Figure 11 is the 'PlayerCamera' class which represents the integration of the Oculus Rift into the game as well as the placement of the physical game camera node within the game environment inside the Player's cockpit. When programming the game the class diagram will prove as a template for the design of the classes within the game and provide guidelines as to class interactions, composition and features.

### 3.4.5 SEQUENCE DIAGRAM

Within the game the player will often engage in a sequence of events that represent the typical 'flow' of gameplay. Figure 12 shows how this typical gameplay flow will progress as the player interacts with the game. From the diagram it can be seen that first the game is initialised, and exists until termination as represented by the leftmost 'Game' object. Whilst the game is running it is possible to initialise a new 'GameState' object by creating a new game, this in turn initialises a 'Gameplay' object that contains the gameplay elements (world, meshes, interface, entities etc.).

Upon initialisation of the 'Gameplay' the 'Player' object is created as well as various 'Enemies'. The Player may damage enemies which record their health until they reach zero in which case they are destroyed and removed from the Gameplay. The same situation may befall the Player where they are destroyed and thus Gameplay itself must end. Here the sequence collapses back until only the 'Game' state remains and may begin once again, or exit entirely.

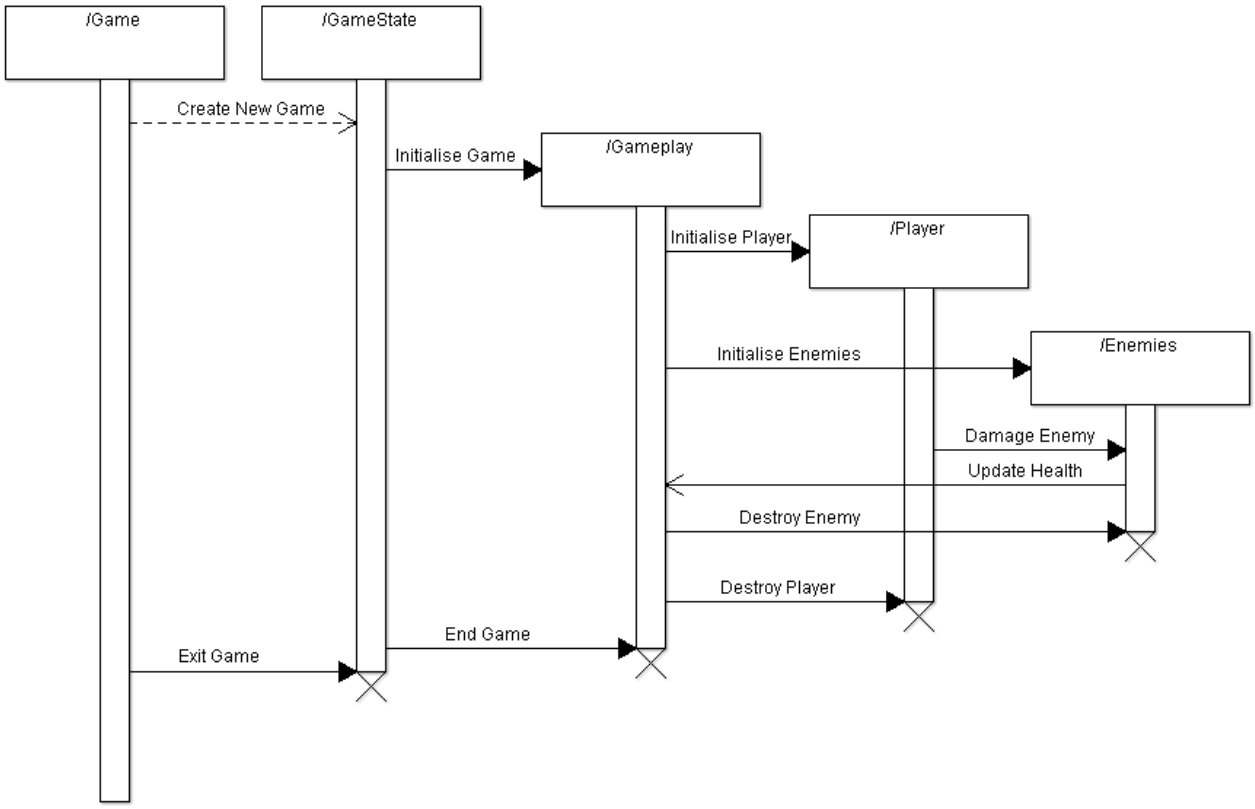


**FIGURE 12 - SEQUENCE DIAGRAM OF GAME EVENTS**

# 4.0 PROJECT PLAN

The overall project plan is best represented in a Gantt chart (Figure 13) as included in the proposal document; it is reproduced here for convenience.

To this date the project is on schedule according to the Gantt chart and all tasks as indicated have been completed up to week 7. Not included on the chart is the research and prototyping that has been completed up until this point, this includes:

- Research into background material for the Oculus Rift itself; this includes technical specifications and background of the technologies and engineering involved in the device, implementations of the Oculus Rift in various demos and engines and experimentation with integration of the Oculus Rift to the Ogre, Unity, Unreal and CryEngine game/rendering engines.
- Research of and experimentation with procedural algorithms for planet generation both using CPU and GPU techniques.
- Research and prototyping of gameplay mechanics in various engines including Unity, Ogre and CryEngine.
- Development of this interim report document including the introduction, objectives, professional considerations, requirements analysis, plan, log and bibliography.

Overall structure of the project can be loosely broken down into several discrete phases of work:

***Phase 1: Proposal/Research Phase*** – The project was researched, chosen and the proposal document presented for review. **Complete.**

***Phase 2: Software Engineering Phase –*** The project was developed and designed from the requirements obtained through analysis of the proposal document and research into similar projects. The outcomes and objectives of the project as a whole were defined and the game was designed to meet these features and requirements. This phase culminates in this interim report.

***Phase 3: Implementation Phase*** – Programming of the game will be begun with the primary objectives in mind and will aim to produce incomplete versions as time progresses with increasing amounts of features:

> *Initial Version* – The first iteration of the project should include the primary elements of the game including a player ship, controls and Oculus Rift head tracking and output.
>
> *Version Revision 2* – The second iteration of the project should expand upon the first and bring weapons, enemies and a basic UI system into the game.
>
> *Version Revision 3* – The third version of the project should include development of the environment/level to create a believable and interesting space scene as well as integrate the remaining gameplay systems.
>
> *Feature-Complete Version* – All primary objectives should be met within this version of the game and it should be considered complete minus the extensions and graphical enhancements.

*Extensions, Bug fixing and Polish* – The final phase of development will focus on enhancing all gameplay elements and including as many extension features as possible. This phase will also concentrate on visuals, the user's experience within the game and additional embellishments and features.

***Phase 4: Testing, Evaluation and Report Phase*** – The final phase will test and evaluate the project and involve completion of the final dissertation document.

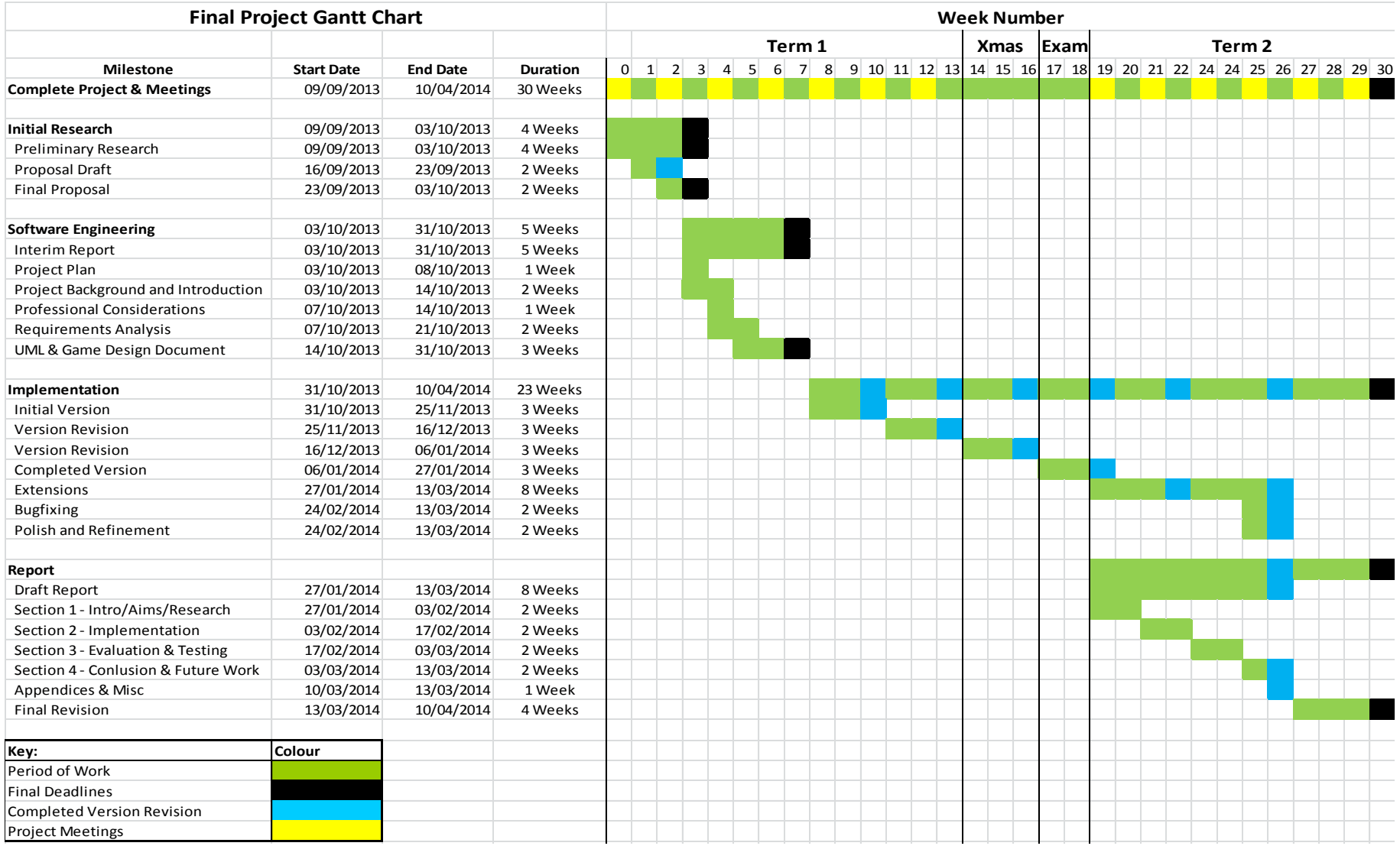


**FIGURE 13 - GANTT CHART OF PROJECT TIMELINE SHOWING PROJECT WORK, DEADLINES AND EXPECTED TIME ALLOCATED TO EACH AND EVERY TASK**

# 5.0 Log of Activities

## 5.1 Interim Log

*Week 0 – 09/09/2013 – 15/09/2013*

- Had initial meeting with project supervisor, discussed project goals and outline as a whole. (30 minutes)
- Began investigating Oculus Rift itself and existing games that utilise the device already. (10 hours)

*Week 1 – 16/09/2013 – 22/09/2013*

- Began initial draft of project proposal. (2 hours)
- Investigated engines for use with Oculus Rift. (8 hours)
- Explored forums, articles and blogs online for impressions in implementing the OR with various games and engines. (6-8 hours)
- Gathered information on matching genres and gameplay types that worked well with the Oculus Rift based on reviews and opinions. (4 hours)
- Experimented with Oculus Rift in CryENGINE 3 SDK, UDK, Torque 3D and Irrlicht Engine. (2 hours)
- Began development of 'space shooter/simulator' idea for project proposal taking into account impressions and experiences with the OR. (2 hours)
- Continued research (approx. 15 hours)

*Week 2 – 23/09/2013 – 29/09/2013*

- Had second meeting with project supervisor, looked at current proposal documents and discussed possible engine usage, as well as refinements to the proposal document itself. (25 minutes)
- Investigated Ogre 3D engine usage with the Oculus Rift. (2 hours)
- Developed a small prototype in Unity, could not get Oculus Rift to work without Pro version and the integration it provides. (2 hours)
- Wrote official project proposal document. (2 hours)
- Developed project Gantt chart, considered layout not well formed for Agile-style development of game product opposed to waterfall project structure, but broad statement of time allocation conveyed in an effective manner. (1 hour)
- Continued research into extensive amount of forums and websites detailing Oculus Rift use, implementation, and similar games in various engines etc. (approx. 20 hours)
- Read chapters 1-10 of *Game Engine Architecture* by Jason Gregory. (approx. 12 hours) [37]

*Week 3 – 30/09/2013 – 06/10/2013*

- Began researching past examples of similar projects and games. Analysed the upcoming releases and noted the recent resurgence of popularity in the space combat simulator genre with examples such as Star Citizen and Elite: Dangerous (4 hours)

- Located various resources on the Ogre forums devoted to similar projects including space environments and space combat simulator gameplay. (4 hours)
- Read more of 'Game Engine Architecture' by Jason Gregory. (10 hours)
- Began refreshing knowledge of C++ (4 hours)
- Researched procedural methods of terrain and planet generation in great detail, implemented a shader-based procedural generator in Unity using a cube-mapped sphere. (20+ hours)
- Explored existing examples of procedural planet generation in Ogre, paid particular attention to previously mentioned cube-mapped spheres as the most suitable method for generating procedural spherical bodies. (10+ hours)

*Week 4 – 07/10/2013 – 13/10/2013*

- Continued to look at extensive procedural planet generation resources. (15 hours)
- Compiled Interim Report template document ready for addition of sections. (1 hour)
- Collated various bibliographical sources using Bib.me web service to aid in management of bibliography which was growing significantly large. (1 hour)
- Remainder of time spent on two other assignments as deadline approached. Limited time available.

*Week 5 – 14/10/2013 – 20/10/2013*

- Initial period of week spent on other projects for deadlines limited time available.
- Continued research into procedural generation. (10 hours)
- Produced introductory section for Interim Report. (4 hours)
- Professional considerations section. (1 hour)
- Requirements analysis section. (5 hours)
- Remaining sections of Interim Report. (5 hours)
- Continued background reading and miscellaneous research. (10 hours)

*Week 6 – 21/10/2013 – 27/10/2013*

- Interim report initial draft reviewed and expanded upon to include suggested enhancements and incomplete features/sections. (2 hours)
- Work on interim report and design continued and expanded upon until complete. (5 hours)
- Read *Evolving Patch-based Terrains for use in Video Games* [38], *Progressive Meshes* [39] (4 hours)
- Read more of *Game Engine Architecture, Real-Time Rendering* [40] and *Effective C++, Third Edition* [41] (10 hours)

*Week 7 – 28/10/2013 – 3/10/2013*

- Interim report submitted (Total editing time 20.45 hours)
- Initial work on project implementation prepared for and begun.

## 6.0 BIBLIOGRAPHY

[1] *Oculus Rift - Virtual Reality Headset for 3D Gaming | Oculus VR* [Online] oculusvr.com, Oculus VR, 2013. Available from: http://www.oculusvr.com/ (Accessed October 2013)

[2] *Early Oculus Rift Reviews from Two Experts* [Online] Road To VR, Lang, B., 2013. Available from: http://www.roadtovr.com/early-expert-oculus-rift-review-dev-kit/ (Accessed October 2013)

[3] Firth, N. *First wave of virtual reality games will let you live the dream*, New Scientist 218(2922), pp 19-20 2013

[4] *Motion Sickness - Causes and Possible Solutions* [Online] Oculus Developer Forums, Oculus VR Developers, 2013. Available from: https://developer.oculusvr.com/forums/viewtopic.php?f=20&t=804 (Accessed October 2013)

[5] *The Internet Sucks: Or, What I Learned Coding X-Wing vs. TIE Fighter* [Online] Gamasutra, Lincroft, P., 1999. Available from: http://www.gamasutra.com/view/feature/131781/the_internet_sucks_or_what_i_.php (Accessed October 2013)

[6] *Reinstall: FreeSpace 2* [Online] PC Gamer, Mahood, A., 2012. Available from: http://www.pcgamer.com/2012/12/22/reinstall-freespace-2/ (Accessed October 2013)

[7] *Strike Suit Zero* [Online] Strike Suit Zero, Born Ready Games, 2013. Available from: http://strikesuitzero.com/ (Accessed October 2013)

[8] *The Making Of: Elite* [Online] Edge Online, Edge Staff, 2012. Available from: http://www.edge-online.com/features/making-elite/ (Accessed October 2013)

[9] *X3: Albion Prelude Info* [Online] EGOSOFT, EGOSOFT, 2013. Available from: http://www.egosoft.com/games/x3ap/info_en.php (Accessed October 2013)

[10] *14 Space games you should be excited about right now* [Online] PC Gamer, Schilling, C., 2013. Available from: http://www.pcgamer.com/gallery/14-space-games-you-should-be-excited-about-right-now/ (Accessed October 2013)

[11] *MobyGames - Chris Roberts* [Online] MobyGames, MobyGames, 2013. Available from: http://www.mobygames.com/developer/sheet/view/developerId,6555/ (Accessed October 2013)

[12] *Star Citizen by Cloud Imperium Games Corporation - Kickstarter* [Online] Kickstarter, Cloud Imperium Games Corporation, 2012. Available from: http://www.kickstarter.com/projects/cig/star-citizen (Accessed October 2013)

[13] *Elite: Dangerous by Frontier Developments - Kickstarter* [Online] Kickstarter, Frontier Developments, 2012. Available from: http://www.kickstarter.com/projects/1461411552/elite-

dangerous (Accessed 2013 2013)

[14] Heaven, D. *Virtual reality rises again*, New Scientist 218(2922), pp 20 2013

[15] *A Critical History of Computer Graphics and Animation* [Online] Wayne Carlson Personal Website - Ohio State University, Carlson, W., 2006. Available from: http://design.osu.edu/carlson/history/lesson17.html (Accessed October 2013)

[16] Seymour, N., Gallagher, A., Roman, S., O'Brien, M., Bansal, V., Andersen, D., Satava, R. *Annals of Surgery - Virtual Reality Training Improves Operating Room Performance*, pp 458–464 2002

[17] NATO Defense Research Group *Virtual Reality,Virtual Reality, Training's Future? Perspectives on Virtual Reality and Related Emerging Technologies*, 1st edn, New York, Plenum Press, (1997)

[18] Zagal, M. *Challenges for success in stereo gaming: a Virtual Boy case study*, In : ACE '09, New York pp.99-106, (2009)

[19] *Oculus Rift Teardown* [Online] ifixit.com, iFixit, 2013. Available from: http://www.ifixit.com/Teardown/Oculus+Rift+Teardown/13682/ (Accessed October 2013)

[20] *Sensor Fusion: Keeping It Simple* [Online] Oculus VR, LaValle:, S., 2013. Available from: http://www.oculusvr.com/blog/sensor-fusion-keeping-it-simple/ (Accessed October 2013)

[21] *The Latent Power of Prediction* [Online] Oculus VR, LaValle, S., 2013. Available from: http://www.oculusvr.com/blog/the-latent-power-of-prediction/ (Accessed October 2013)

[22] *Building a Sensor for Low Latency VR* [Online] Oculus VR, Oculus VR, 2013. Available from: http://www.oculusvr.com/blog/building-a-sensor-for-low-latency-vr/ (Accessed October 2013)

[23] Olson, J. L., Krum, D. M., Suma, E. A., Bolas, M. *A design for a smartphone-based head mounted display* , In : Virtual Reality Conference (VR), 2011 IEEE , Singapore pp.233-234, (2011)

[24] *Oculus Rift: Step Into the Game by Oculus* [Online] Kickstarter, Oculus VR, 2012. Available from: http://www.kickstarter.com/projects/1523379957/oculus-rift-step-into-the-game (Accessed October 2013)

[25] Freie Universität Berlin. *Global Computer Vision*, Available from: http://robocup.mi.fu-berlin.de/buch/chap9/ComputerVision.htm (Accessed October 2013)

[26] *What Does It Look Like In the Oculus Rift?* [Online] Road To VR, Lang, B., 2013. Available from: http://www.roadtovr.com/what-does-it-look-like-in-the-oculus-rift/ (Accessed October 2013)

[27] *Digital Foundary: Hands-on with Oculus Rift* [Online] Eurogamer.net, Leadbetter, R., 2013. Available from: http://www.eurogamer.net/articles/digitalfoundry-hands-on-with-oculus-rift (Accessed October 2013)

[28] *Oculus Rift* [Online] Oculus VR, Oculus VR, 2013. Available from: http://www.oculusvr.com/ (Accessed October 2013)

[29] *About OGRE - Open Source 3D Graphics Engine* [Online] OGRE 3D, Torus Knot Software Ltd, 2013. Available from: http://www.ogre3d.org/about (Accessed October 2013)

[30] *Game Physics Simulation* [Online] bulletphysics.org, Game Physics Simulation, 2013. Available from: http://bulletphysics.org/wordpress/ (Accessed October 2013)

[31] *libnoise: a portable, open-source, coherent noise-generating library for C++* [Online] libnoise, Bevins, J., 2007. Available from: http://libnoise.sourceforge.net/ (Accessed October 2013)

[32] *Boost C++ Libraries* [Online] Boost C++ Libraries, Dawes, B., Abrahams, D., 2007. Available from: http://www.boost.org/ (Accessed October 2013)

[33] *Oculus Rift SDK* [Online] Oculus VR Developer Center, Oculus VR, Inc, 2013. Available from: https://developer.oculusvr.com/?action=dl&p=sdk&v=8 (Accessed October 2013)

[34] *Code of Conduct for BCS Members* [Online] British Computer Society, British Computer Society, 2011. Available from: http://www.bcs.org/upload/pdf/conduct.pdf (Accessed October 2013)

[35] *Designing around motion sickness* [Online] Oculus Developer Forums, Oculus Developer Forums, 2013. Available from: https://developer.oculusvr.com/forums/viewtopic.php?f=32&t=15 (Accessed October 2013)

[36] *6DOF* [Online] Wikimedia.org, Ionescu, H., 2010. Available from: http://upload.wikimedia.org/wikipedia/commons/f/fa/6DOF_en.jpg (Accessed October 2013)

[37] Gregory, J. *Game Engine Architecture*, 1st edn, Natick, Ma, USA, A K Peters, Ltd., (2009)

[38] Raffe, W., Zambetta, F., Li, X. *Evolving Patch-based Terrains for use in Video Games*, In : GECCO '11, Proceedings of the 13th annual conference on Genetic and evolutionary computation pp.363-370, (2011)

[39] Hoppe, H. *Progressive meshes*, In : SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive technique, New Orleans, USA, vol. 23, pp.99-108, (1996)

[40] Möller, T., Haines, E. *Real-time rendering*, 2nd edn, Natick, Mass, AK Peters, (2002)

[41] Meyers, S. *Effective C++, Third Edition, 55 Specific Ways to Improve Yout Programs and Designs*, 3rd edn, Westford, Mass, Pearson Education, (2005)

# 7.0 APPENDIX

## 7.1 PROPOSAL DOCUMENT

## 7.2 Game Design Document