MSc in Computer Games Technology

Project Report

2014

A skiing game utilising Oculus Rift head tracking to intuit body position and infer slalom skiing movement

Mark Young

mark_young@hotmail.com

Supervised by Dr. Greg Slabaugh

10th January 2014

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed:

Mark Young

Abstract

This project report describes the “design and build” process undertaken to create a slalom skiing game using the Oculus Rift. The Oculus Rift is an exciting new way to experience games. It is a virtual reality head mounted display with three-axis movement sensors that track the player’s head position enabling the game world to be rendered to match their head orientation. The game is viewed through the eyes of the player’s avatar skiing down a slalom course. Movement control is achieved by sensing the tilt of the player’s head as they lean from side to side steering from gate to gate.

Key words

Oculus Rift, Slalom Skiing, Head Tracking, Software Engineering, Game Design

Acknowledgements

I would like to thank my project supervisor Dr. Greg Slabaugh for his guidance and wise words throughout this project; my friends and family, especially my wife, for their support and understanding when I absented myself from social engagements; the managers at Snow & Rock, Chertsey for listening to my “weird and whacky” proposal and then allowing me to spend a day in their store letting their customers try my head mounted skiing game, and the staff for being so accommodating (and making me coffee) and Mr Sal Gunduz for being my assistant, and finally all of the participants of the evaluation for providing their feedback on the experience.

Table of Contents

1 Introduction 8
1.1 Motivation 8
1.2 Research questions 9
1.3 Aims and Objectives 9
1.4 Work plan 10
1.5 Specific Goals 10
1.6 Beneficiaries 11
1.7 Project Outcomes 11
1.8 Major Changes 12
1.9 Report Structure 12
2 Academic Context 13
2.1 Background 13
2.2 Head mounted displays 15
2.3 Other hardware enhanced virtual reality skiing games 16
2.4 Software Engineering 17
2.5 Spiral development model 18
2.6 Game Architecture 19
2.7 Development References 21
3 Methods 23
3.1 Development Method 23
3.2 Development Environment 23
3.3 Version control, backups and tools 24
3.4 Mood board & original concept drawing 25
3.5 Game Design 27
3.5.1 Summary of the game 27
3.5.2 Game play 27
3.5.3 Game progression 27
3.5.4 Player controls 27
3.5.5 Difficulty system 28
3.5.6 In-game UI 28
3.5.7 Menu options and navigation 29
3.5.8 Saving/Loading system 29
3.5.9 Interaction matrix 30
3.5.10 Collision detection 31
3.5.11 Collision response 31
3.5.12 Scoring 31
3.6 Design Patterns 31

3.6.1 Singleton & Factory Design Patterns 31
3.6.2 State Pattern (Finite State Machine) 32
3.6.3 Observer Pattern (Publisher-Subscriber) 33
3.7 Terrain acquisition 34
3.7.1 Find terrain in SketchUp Pro 35
3.7.2 Viewing Terrain in SketchUp Pro 36
3.7.3 Unity Terrain 37
3.7.4 Unity Terrain mesh 38
3.7.5 Terrain populated 39
3.7.6 Course creation 40
3.7.7 Final view 41
3.8 Testing 41
3.9 User study 42
3.9.1 Practise evaluation 42
3.9.2 Real evaluation 43
3.9.3 Recording the results 43
3.9.4 Evaluation form 45
3.9.5 Feedback from the general public 47
4 Results 51
4.1 Images of the game 51
4.2 Images of the game evaluation day 55
4.3 Videos of the game 57
4.4 Speed and latency 57
4.5 'Look to select' menu system 58
4.6 UML high level design diagrams 60
4.7 Barrel distortion 61
4.8 Entering the player's name into the high score table 62
4.9 Marking the sides of the course 63
4.10 Lighting and shadows 63
4.11 Physics 66
4.12 Levels 70
4.13 Summary of key results 70
5 Discussion 72
5.1 Overall objectives 72
5.2 Prototyping using Unity vs. C++ 74
5.3 Speed control via crouching 74
5.4 Evaluation 75
5.5 A question of disorientation 75
5.6 Steering 76
5.7 Recommendations 77

5.8 Comparison to Other Work 78
5.9 Summary of lessons learnt 78
6 Evaluation, Reflections and Conclusions 80
6.1 Project Management 80
6.2 Future work 80
6.3 Reflection 81
6.4 Conclusion 81
7 References 82

Table of Figures

Appendix A: Project Proposal for MSc in Computer Games Technology A1
Appendix B: Example code B1
Appendix C: Use Case Specifications C1
Appendix D: Menus D1

Table of Figures

Figure 1 - Virtual Reality Skiing Simulator 14
Figure 2 - Spiral development model 18
Figure 3 - Adjusting for Eye Offsetting 20
Figure 4 - Mood board 25
Figure 5 - Original concept drawing 26
Figure 6 - Planned in-game GUI layout 28
Figure 7 - Planned game menu structure 29
Figure 8 - Interaction Matrix 30
Figure 9 - Player finite state machine 32
Figure 10 - Google Earth imagery in SketchUp Pro 35
Figure 11 - Google Earth Terrain in SketchUp Pro 36
Figure 12 - Terrain heightmap mesh in Unity 37
Figure 13 - Unity Terrain with heightmap applied 38
Figure 14 - Unity terrain with trees 39
Figure 15 - Unity course design 40
Figure 16 - Player view of course finish line 41

Figure 17 - User evaluation responses spreadsheet 43
Figure 18 - User evaluation summary spreadsheet 44
Figure 19 - User evaluation summary spreadsheet formulas 44
Figure 20 - User evaluation spreadsheet verification cells 44
Figure 21 - User evaluation questionnaire 46
Figure 22 - Feedback results 48
Figure 23 - View from start line 51
Figure 24 - Skiing, approaching a slalom gate 52
Figure 25 - Real-time detailed tree shadow across the course 52
Figure 26 - Making a sharp turn into a gate 53
Figure 27 – Jumping onto rails in the Terrain Park 53
Figure 28 - Approaching the finish line 54
Figure 29 - Close up of the in game UI 54
Figure 30 - User testing at Snow & Rock, Chertsey 55
Figure 31 - More pictures from the user evaluation day 56
Figure 32 - Frame rate distribution 58
Figure 33 - 'look to select' the Val Thorens menu option 59
Figure 34 - User view of menu system 60
Figure 35 - UML - Use Case diagram 60
Figure 36 - UML - Class Diagram 61
Figure 37 - Unprocessed image prior to barrel distortion 62
Figure 38 - Smooth blending of baked and real-time shadows as camera approaches 64
Figure 39 - Poor shadow handling as camera approaches 65
Figure 40 - Skier collider object 1 – rectangle around skis 66
Figure 41 - Skier collider object 2 - huge plank 67
Figure 42 - Skier collider object 3 – twin colliders 68
Figure 43 - Skier about to fall over 69
Figure 44 - Player view when fallen over lying on side 69
Figure 45 - Nine courses on three terrains 70
Figure 46 - 2D GUI indicator to show direction of travel 76
Figure 47 - Source code - GameManager.cs B1
Figure 48 - Source code - MenuItemController.cs B3

1 Introduction

This is a design and build project fulfilling the Individual Project part of the MSc in Computer Games Technology at City University, London. The aim was to create a slalom skiing game using the newly invented Oculus Rift (OculusVR 2013). The Oculus Rift is an exciting new way to experience games. It is a virtual reality head mounted display with movement sensors that is due to be available for consumer release in 2014. This means that the player can be immersed into the game world such that the view presented to them aligns with their head orientation.

The principle aim of this project was to put the player into a first person perspective skiing down a mountain utilising the Oculus Rift's head tracking ability to allow the player to look around. A secondary goal was to attempt to intuit the player's physical position and apply that to the in-game character movement.

There is much to learn about the requirements of human perception whilst wearing a head mounted display. The Oculus Rift SDK overview document (*Oculus SDK Overview v0.2.1* 2013) takes pains to state that for the best virtual reality experience, the frame rate must hold consistently at 60fps with 'motion-to-photon' latency of no more than 60ms and ideally less. These are critical factors in achieving an immersive experience.

1.1 Motivation

This project was motivated by the opportunity to work with brand new technology that opens up a new dimension in game play experiences. When this project started the Oculus Rift had not yet been made publically available, consequently it was a new and challenging field. No books or tutorials were available and only a limited knowledgebase of fellow developers. By the time the developer model arrived very few games had been made for the Oculus Rift. This project describes the first (to our knowledge) skiing game developed for a head mounted display such as the Oculus Rift.

The developer model Oculus Rift has been made available before the consumer quality model to enable developers to get hands on experience with the technology and start exploring its boundaries. The consumer model is expected to have a higher resolution screen. The full specification has not been released.

This project explores hardware-assisted game play and aims to expand the field by developing a skiing game that is fully operable via the head mounted display. A new menu system is developed that introduces a 'look to select' mechanism whereby the user selection is indicated by detecting the user staring at their desired menu option.

1.2 Research questions

There are several research questions underlying this project, including:

1. Is it possible to design and build a real-time skiing game using the Oculus Rift where the player controls the skier through head motion?
2. For such a game, what is the user experience in terms of immersion, responsiveness, and fun?
3. For such a game, will the player experience disorientation and/or motion sickness?
4. Is it possible to create a menu system controlled from the view inside the Oculus Rift that does not require a mouse or keyboard?
5. For such a menu system, does the user find the mechanism intuitive?

1.3 Aims and Objectives

To address the research questions, the project has the following objectives:

	Objective	Priority	Testable Result
1	A literature review of hardware enhanced skiing games and head movement player controls	High	Inclusion into the Project Report
2	3D scene rendering to provide a first-person view that changes based on head movement	High	Images in Project Report and video of game in action
3	Player control of skier using head tracking input to control speed and make slalom turns	Medium	Video of game being played
4	A consistent frame rate of 60 frames per second (FPS)	Medium	Benchmarked using FRAPS (Fraps 2013)
5	User evaluation including, but not limited to: immersion; responsiveness; movement control feedback; motion sickness and fun	High	Feedback from players recorded and summarised in Project Report

1.4 Work plan

This project included a number of technical challenges and involved working with the Oculus Rift at a time when development resources were fluid making it more challenging to evaluate which language and approach to use to code the game. An early evaluation of the documentation and SDK was essential to decide between the use of C++ using DirectX or OpenGL, and Unity. At that time the software development kit (SDK) contained sample code written for DirectX only, whilst this author's experience is with OpenGL. The Oculus Unity integration package had just been released, so the prudent plan was to use Unity to produce a prototype of the game and look to adopt OpenGL later if feasible or necessary.

The complete Gantt chart can be found at the end of the proposal document in Appendix A. Research and review of relevant literature occurred first informing the development work. Major development milestones were:

Milestone	End Date	Description
1	07/08/2013	Generation of static scene
2	27/08/2013	Ability to ski down slope
3	21/09/2013	Create a virtual reality view
4	21/10/2013	Slalom skiing based on head position
Final	31/10/2013	Round out the game

1.5 Specific Goals

1. Prototyping using the Unity engine for the best chance at producing a viable sample game.
2. If time permits, a C++ and OpenGL version of the game will be built. The main reason to adopt this path will be for performance and ultimate developer control.
3. A game design will be produced to document the expected player experience.
4. The game will be built using a Model View Controller (MVC) loosely coupled flexible game architecture.

5. An iterative approach will be taken to the development of the skiing game and the head mounted control system. The software will be fleshed out in stages including:
    a. Static scene generation
    b. Movement down a slope
    c. Sensor input from Oculus Rift to map view to head orientation
    d. Oculus Rift sensor input to define speed of movement and sharpness of turns
    e. The addition of gameplay mechanisms to encourage repeated play of the game such as quickest time, highest score and fastest speed achieved, with associated leader boards to encourage players to compete.
6. An evaluation of player experiences will be conducted, ideally via the cooperation of a local skiing shop or dry ski slope. My intention is to approach the shop and see if they will let me set up the Oculus Rift and slalom skiing game on their premises for a day. Adult members of the public will be encouraged to fill in a questionnaire and provide feedback on their experience.

1.6 Beneficiaries

The primary beneficiary will be the author of this project who will gain wide ranging experience in games development using new technology and incorporating that with both Unity and potentially OpenGL with C++. This project report and the game itself will also be a valuable portfolio element. Additional beneficiaries will be players of this game, which may include complete ski beginners getting a feel for skiing. Finally lessons learned from this project may be informative to other developers considering similar work using the Oculus Rift. One way in which the experiences from this project will be shared with others is via the Oculus Developer forum website.

1.7 Project Outcomes

The primary outcome from this project is the slalom game and the user evaluation study of it. Included in that is the user acceptance of the experimental menu based on a ‘look to select’ system.

1.8 Major Changes

The original planning for this project included the possibility that the game might be coded in C++, utilising either DirectX or OpenGL, however the Unity game engine proved to be very capable and thus there was no need for an alternate approach.

The original menu structure for the game was overly complex and impractical for use within the Oculus Rift headset. It was hugely simplified and redesigned during the implementation phase of the project.

1.9 Report Structure

Chapter 1 – Introduction - provides an introduction to the project, its goals, objectives and planned outcomes.

Chapter 2 – Academic Context - looks at the background to this project and reviews the key literature related to the project, and how that informs the choices made in this project.

Chapter 3 – Methods – the development method is described in addition to a full game design and a description of some design patterns deployed and the techniques used to build a game level. Finally the user evaluation approach is discussed.

Chapter 4 – Results - looks at the game produced and the results from the user evaluation of it. Some of the interesting implementation challenges are examined in detail.

Chapter 5 – Discussion - reviews the original objectives against the final game produced and details the extent to which they have been achieved. Details are provided where the final project differed from the project plan and some key lessons learnt and recommendations are provided.

Chapter 6 – Evaluation, Reflections and Conclusions – the entire project is evaluated, conclusions drawn and the experience gained from developing the game summarised. Finally some proposals for further work are made.

Beyond these chapters a copy of the original project proposal can be found in Appendix A and a couple of the C# game code scripts are listed in Appendix B, the full Unity project directory and a compiled version of the game can be found on the accompanying CD along with a photo of every completed evaluation form

2 Academic Context

This literature review aims to provide a broader context for this project. First the history of the computer game industry is examined with a focus on games differentiating themselves through the use of specialist hardware. Then existing research performed on the specific challenges of head mounted displays is summarised, leading to a review of other hardware-assisted virtual reality skiing games. Finally software engineering and architecture for games is looked at along with a review of the available Unity development books.

2.1 Background

In the late 70's and 80's computer games came into the public consciousness in the form of arcade machines and a few notable films. At that time home computing was not able to offer a comparable experience. In the following decades successive generations of games consoles became accessible to a wide audience forcing arcade machine manufacturers to strive to offer experiences that players could not get at home. This led to many hardware innovations and more physical interactions in the form of games such as

- Hard Drivin' – 1989 – Unforgiving driving simulation (IAM 2013a)
- Sega R360 – 1992 - 360 degree flight game (Broyad, T 2013a)
- Ridge Racer Full Scale – 1994 – Player sat in Mazda MX5 playing in front of an 18' screen (Broyad, T 2013b)
- VR-1 – 1994 – 32 players wearing head mounted displays, space shooting game (Broyad, T 2013c)
- Final Furlong – 1997 - Horse rider / jockey game (IAM 2013b)
- Dance Dance Revolution – 1998 – Player dances by stepping on pads in time with music (Broyad, T 2013d)
- Police 24/7 – 2001 – Tactical shooter, player stance tracked. Physically tiring (IAM 2013c)
- Further examples of the wide range of arcade cabinet designs can be found online (RoG et al. 2013)

Home based systems continued to chase the quality of experience that could be provided in the arcades. Unique and innovative controls came to the home based systems: Dance mats, Musical controllers for guitars and drums, the PlayStation Eye Toy (2003) was a webcam like device that enabled the console to recognise gestures and body posture, the Wii Controller (2006) created a resurgence of interest in physical interaction with games and Microsoft's Kinect Sensor (2010) which consists of

a camera and voice recognition with a UI overhaul enabling players to control the Xbox 360 and play games without using a physical controller at all. With devices like these video gaming is becoming less sedentary and more fun, not just for the players taking part but also for the other family members who can laugh along at their crazy antics in front of a high energy game.

Virtual reality is a widely used term that ranges from the sit in car experience of Ridge Racer Full Scale through the body position recognition in Police 24/7 culminating eventually with the full immersive experience seen depicted as the Holodeck in Star Trek which is currently being recreated by a team at the University of Southern California utilising consumer grade technology including the Oculus Rift (Project Holodeck 2013). The Oculus Rift promises to further reduce the gap to immersive virtual reality with its head mounted display. Imagine the opportunity to play Alpine Racer 2 (IAM 2013d) wearing an Oculus Rift with a fan blowing chilled air towards your face.

Figure 1 - Virtual Reality Skiing Simulator (Untouchable Events 2013)

Figure 1 shows a lady playing Alpine Racer 2. This is a stylised image since the original cabinet's monitor has been replaced with a floating screen. She is shown wearing unidentified glasses, perhaps 3D although Namco's game did not support a 3D display.

Perhaps on the back of the Oculus Rift's highly popular kickstarter campaign other new virtual reality devices are gaining momentum. One particularly promising looking device is The Omni by Virtuix (Virtuix 2013). It is a controller on which the player stands and can then walk, run and jump such that their in-game avatar performs the same movements. It incorporates a supporting waist belt so that the player can be using an Oculus Rift at the same time. Virtuix has raised more than five times its kickstarter funding goal in just one third of its funding period, showing that there is a lot of interest in having a home based virtual reality experience.

2.2 Head mounted displays

Head Mounted Displays (HMDs) have many applications including fighter pilot helmets providing additional tactical data; police and firefighters viewing thermal images and maps whilst also seeing through to the real world; surgeons being shown a combination of computer imagery such as CT scans and MRI imagines along with the patients vital signs. These are typically augmented reality (AR) displays, such that the wearer is seeing the real world along with additional computer generated displays.

Use of a head mounted display such as the Oculus Rift for a computer game replaces the real world with the game world. This can lead to disorientation and motion sickness as has been demonstrated by Omar Merhi et al in their paper 'Motion sickness, console video games, and head-mounted displays' (Merhi et al. 2007). Other studies (Peli E, 1998) have looked to see if a HMD could be harmful to the human visual system and concluded that despite earlier studies to the contrary "no harmful nor medically significant long term changes were observed". In fact far from being viewed as dangerous some medical researchers are using HMDs in their research for instance in the rehabilitation of stroke victims (Lee et al. 2012) and the treatment of the eye conditions strabismus, also known as cross-eye, and amblyopia, also known as a weak or lazy eye, through the playing of a 3D game in the Oculus Rift (Blaha, J 2014).

Given that the Oculus Rift itself is brand new technology, albeit built on top of a wealth of previous head mounted displays and sensors, specific research into the Oculus Rift itself is too new to have made it into published papers, consequently the literature related directly to the Oculus Rift has been drawn from conferences and developer blogs. For instance at 2013's Game Developers Conference Michael Abrash from Valve gave a talk highlighting the limitations of current virtual reality (VR) technology and positing that we are a number of years, perhaps decades, away from delivering a completely believable VR display system (Abrash, M 2013a), despite the Oculus Rift being a good start that arrives at a time when many converging technologies come together to make it available at consumer prices. He describes how current display systems typically light pixels for a set period of time per frame, in some cases each colour (red, green and blue) for a third of a frame each, and when that is combined with head movement, and indeed eye movement, this leads to image lag. He demonstrates this using high speed cameras and slow video playback, at normal display rates these anomalies are seen as a blur. Unfortunately the "human perception system" (as he terms it) seeks to identify anomalies, perhaps from our ancestry of competing in the food chain both as predator and prey, and so it appears that we are hardwired to notice such visual inconsistencies. In addition to his talk at GDC, Michael Abrash continues to publish a fascinating series of articles focusing in

depth on visual perception and VR/AR challenges, they make extremely good reading (Abrash, M 2012, 2013b, 2013c, 2013d).

John Carmack, founder of Id Software, has written an excellent article entitled 'Latency Mitigation Strategies' in which he incrementally breaks down the rendering process and suggests strategies for taking multiple sensor samples per frame whilst overlapping the rendering and GPU drawing process, finally warping the resultant image during the video scan-out process with additional sensor samples (Carmack, J 2013). This last technique is reminiscent of Michael Abrash's descriptions of 'racing the beam' where again the full screen scene is produced in smaller blocks just ahead of the hardware buffer reading process (Abrash, M 2012).

In this project the game will be created without advanced graphical enhancements such as blur and depth of field as the literature review has revealed that these techniques are more likely to cause noticeable visual artefacts. However it is beyond the scope of this project to attempt to adopt more advanced techniques such as 'racing the beam'. Instead, the user evaluation study will ask users to comment on their feelings regarding motion sickness and disorientation.

2.3 Other hardware enhanced virtual reality skiing games

As has been mentioned earlier one of the most compelling early examples of a virtual reality skiing game was Alpine Racer in 1995 and Alpine Racer 2 (IAM 2013d) in 1996 produced by Namco. It featured a unique controller, a pair of steps on which the player stood like a pair of skis. Swinging the steps from side to side caused turning whilst the foot position could be tilted to create tighter turns similar to the way a skier uses their edges. The player was positioned about a meter away from a 52" projection monitor, which helped to make the game more immersive. Namco did release Alpine Racer 3 for the PlayStation 2 but sadly this was before the Eye Toy had been released and thus the game was played with a standard controller. In 2009 Namco Bandai released a version for the iPhone which used the accelerometer to detect the phone being tilted and thus steer the player.

In 2008 Nintendo released Wii Fit which came with the Wii Balance Board. The Balance Board contains four pressure sensors and is able to accurately track the balance and weight distribution of the user (Clark et al. 2010). This made it an ideal controller for a skiing or snowboarding game. Wii Fit included a Slalom Skiing mini game that was able to sense the player's weight shifting left and right for turns and also leaning forward to increase speed. There were a number of third party skiing and

snowboarding games released that also utilised the Balance Board for player control including Namco Family Ski & Snowboard.

In 2011 Microsoft published Kinect Sports: Season Two to be played with its Kinect body posture tracking system. This included a skiing game which recognised the player's body position to detect leaning left and right for steering, crouching for a speed boost and jumping.

Previous research has been performed to recognise and react to physical body position for the purposes of virtual skiing. In 2008 an art installation was built in Ljubljana, Slovenia to allow a user to "immerse himself into the skiing sensation without using any obvious hardware interfaces" (Solina F, Batagelj B, Glamočanin S 2008). This was done by creating a physical ski slope scene, complete with artificial snow, skis and poles for the user to use, in a room facing a wall onto which a virtual ski slope scene is projected. A small camera in front of the user monitored body posture and used this to drive in game movement: left and right turns and crouching to increase speed. This approach to virtual skiing focused highly on computer vision to determine skier movement whereas this project aims to envelope the player in the game world and utilise head position to control player movement.

Companies like SkyTechSport (SkyTechSport 2013) are combining high tech sports training equipment with huge virtual reality projected displays to enable athletes to train on GPS mapped replicas of Giant Slalom race courses in preparation for major events.

The aim of this project is to combine and build on the best aspects of these hardware-assisted skiing games. The Alpine Racer 2 arcade game and the Sky Tech Sports system gave a sense of immersion due to the large screen close to the player and Kinect Sports tracked body posture to control the skier. This project will provide a more immersive experience utilising the Oculus Rift and will attempt to intuit body posture from the head movement of the player.

I have not been able to find a previous attempt to make a head mounted display virtual reality skiing game, which makes this a very exciting project with the possibility to break new ground.

2.4 Software Engineering

Software engineering aims to formalise and improve the process of software development through the use of proven practices. In their book "Game Architecture and Design" (Rollings & Morris 2004, p. 98) argue that game development is not well suited to the traditional waterfall method of software design because "much of

gameplay is emergent, in that you don't know quite how all those rules will work in practice until you try them out", they go on to say that games are best built using an iterative process such as the spiral development model because this allows for regular assessment of unanticipated things. Although this project involves a single person the spiral model will be used to ensure that development is focused on achieving easily defined and measurable goals that increase in complexity leading towards the final game.

2.5 Spiral development model

The spiral development model (INM375 2012, lecture 2, slide 38), shown in Figure 2, repeatedly iterates over the central development stages, with risk analysis being a crucial factor in determining the process model for the project.

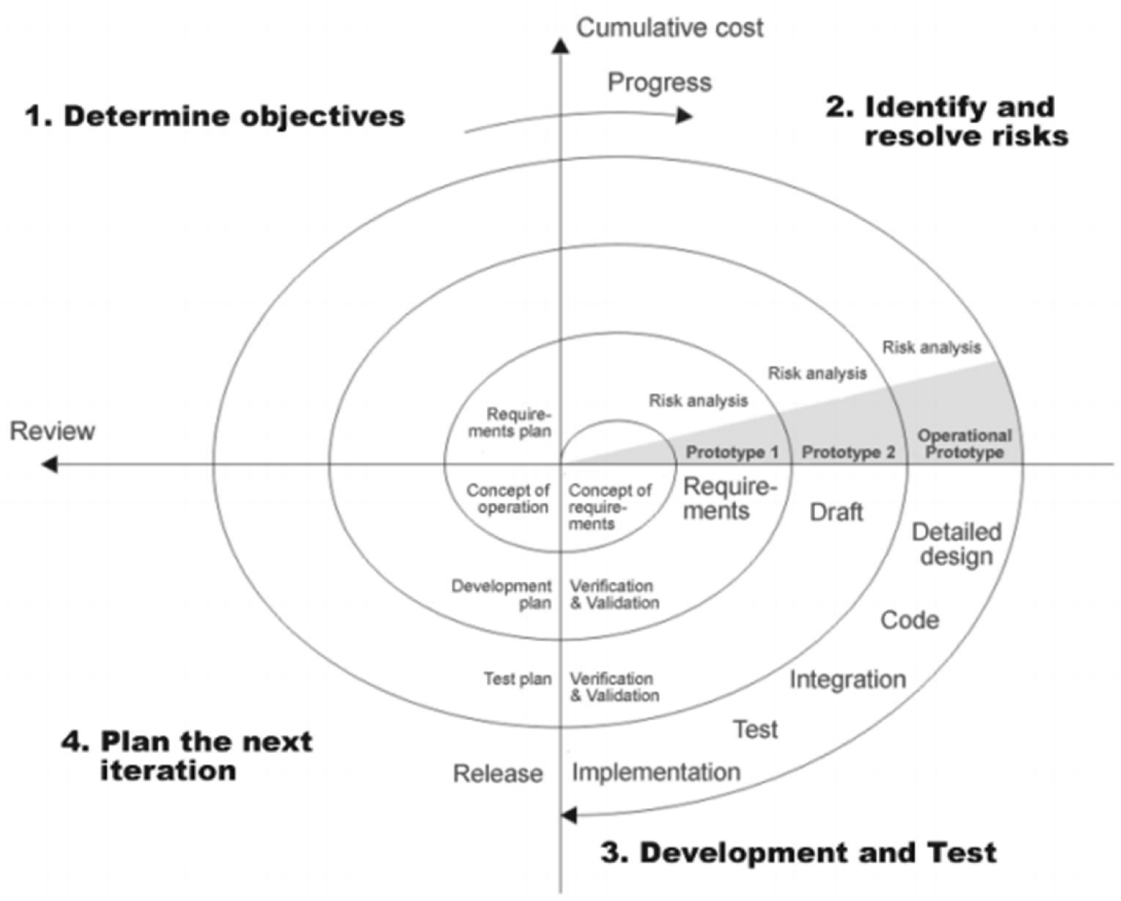


Figure 2 - Spiral development model

In this project there are no external stakeholders so the risk analysis will be done by the developer by aiming to deliver each milestone build on time according to the Gantt chart. The milestones are summarised in section 1.4.

2.6 Game Architecture

Designing a game around a solid game architecture is core to the production of high quality software. Whilst independent (indie) game development is enjoying a resurgence in recent years due to the availability of online discovery and distribution systems such as Steam (Valve 2013), in the games industry games are getting bigger and bigger requiring dozens if not hundreds of professionals to coordinate and contribute to the final production, often comprising of teams in geographically remote locations, sometimes also involving a language barrier. In some cases different aspects of the game are designed and built by different teams, this doesn't always work well, for instance in the game Deus Ex: Human Revolution the boss battles were outsourced to another company that failed to understand the importance of the stealth and choice mechanics of the main game and so produced a series of standard shooter boss battles which were perceived by many critics as incongruous.

Game engine architectures should be as flexible as possible. Even if the class hierarchy is architected perfectly for the first game, sequels or other variant games will likely come along later that will cause tension on the original design and restrict or hamper code reuse. C++ style class hierarchies can become burdensome over time if they are continually added to, as often happens over the life time of a game engine. Scott Bilas of Gas Powered Games gave a talk at GDC 2002 (Bilas, S 2002) where he described code as having a tendency to "harden" over time especially as class hierarchies get deeper and more convoluted. A good example of this can be found in Mick West's article Evolve Your Hierarchy (West, M 2007). These papers instead suggest building games around a component based game object system. This literature directly informed the decision to select Unity3D for this project since it is a component based game object system.

The Unity3D game engine (Unity Technologies 2013) is a fully integrated development environment (IDE) that allows one to assemble art and assets, called GameObjects, into 3D scenes, and then apply components to each GameObject to give it additional characteristics such as lighting, audio, special effects, physics, animation and scripting. This creates a very flexible system and avoids the sometimes difficult re-factoring that can occur when building deep class hierarchies of game elements.

Software architecture patterns such as Model-View-Controller (MVC) separate out the game into three distinct components: Model – The data structures that model the entire game world; View – The visual representation of the game world presented to the player and Controller – The processing of input from the player which will feed into updating the Model. It is good practise to build an event system between these three components allowing the Controller to send commands to the Model to update

its state based on user input; the Model may notify the View that there has been a change in state and the View will request information from the Model to generate its output.

Another facet of game architecture is the design of the game itself, how it is visually presented and how the user interfaces with it. The best Oculus Rift games must take account of these additional areas. Nick Whiting, a senior programmer at Epic Games, working on Unreal Engine 4, wrote of his experiences integrating the Oculus Rift into UE4 (Whiting, N 2013), he points out that beyond the latency issue one also needs to pick the right art style to best suit the head mounted LCD display in the Rift, and choose which advanced graphic options to use, or more specifically to avoid, such as Depth-of-Field and Motion Blur, which don't work well on an HMD display. It's also best to model physical head movement as accurately as possible in game, for instance when looking at ones toes the head pivots around the neck and the eyes follow a downward arc. They don't rotate around the centre of the head, as shown in Figure 3.

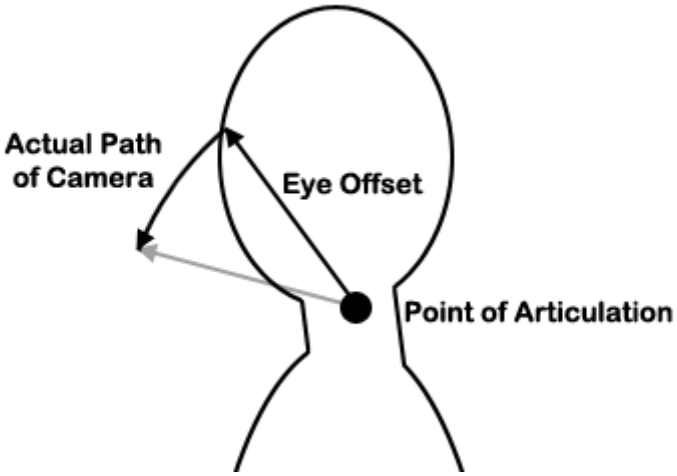


Figure 3 - Adjusting for Eye Offsetting

Given this advice the game design and implementation of this project was kept to a minimalist graphic style. The camera movement was designed to be as natural as possible to match physical eye position and movement, for instance no camera fly over the course before starting a level. Likewise the menu system was built as objects in the game world rather than a list of text choices on a blank screen. This was to avoid jarring transitions between the game world and the menu.

2.7 Development References

Book publishing struggles to keep up with fast moving products like Unity. At the time of the literature review phase of this project Unity v4.0 had been released for seven months and only one Unity 4 book had been published. *Unity 4.x Cookbook* (Smith & Queiroz 2013) contains an excellent collection of recipes and instructions covering some interesting topics such as performing homemade motion capture in Unity using a Microsoft Kinect, creating online leaderboards and implementing 3D using a pair of projectors with polarized filters, although the Oculus Rift completely replaces and obsoletes that particular approach. This is much more of an intermediate level book that assumes the reader knows the basics of Unity and its interface.

Too late for a full literature review in this project *Beginning 3D Game Development with Unity 4 2nd Edition* (Blackman, S 2013) has recently been published. The first edition of this book was written for Unity 3 and from the reviews on Amazon it would appear that it has not been fully updated for the changes in Unity 4, thus I would not recommend this book. Also *Unity 4.x Game AI Programming* (Kyaw & Swe 2013) is now available and appears to be reasonably well received from the limited reviews it has on Amazon. It covers the basics of AI such as Behaviour Trees, Finite State Machines, Path finding, Path following, Flocking and Probability but fails to deliver in-depth detail on Behaviour Tree algorithms and doesn't have a particularly compelling example project to demonstrate the AI techniques it covers. It would appear that this book is probably worth a read provided one's expectations aren't set too high regarding the level of detail provided for the AI topics.

Given the lack of available Unity 4 books, and despite this project being based on Unity 4, it is worthwhile to briefly review those written for Unity 3. First is *Unity 3.x Game Development by Example* (Creighton, R.H 2011), which does a great job of walking the reader step by step through the creation of four different game ideas in a cookbook or follow-along style. It's a brilliant resource for a total beginner at Unity and someone who likes to learn in this fashion.

Game Development with Unity (Menard, M 2012) is another follow-along book. This one includes some useful chapters on some of Unity's tools such as the Terrain Tool which allows one to create a 3D terrain to be used in game, and includes more information about scripting, profiling and debugging, however it is still not much more than an introductory text and does little to conceptualise how one might approach a larger or more sophisticated project.

Unity 3 Game Development Hotshot (Wittaybundit, J 2011), is also primarily a follow-along book. It does have a few very brief appendices which cover slightly more advanced topics such as coroutines and shaders. Coroutines provide the ability to

perform tasks over a longer period of time than per frame, execution can be yielded either until the next frame, until a period of time has expired.

Unity 3.x Scripting (Gerasimov & Kraczla 2012) promised to be a more in depth book focusing entirely on scripting, and includes a brief appendix on object-orientated Programming in Unity. Unfortunately a large number of omissions and inaccuracies in this book render it unusable. The associated files are missing numerous elements such as meshes and scripts, which leaves the reader having to debug the examples and figure out missing instructions for themselves. This makes it more of a trial by fire than a useful teaching aid. I persevered with it for a while and even posted some of the solutions I found to making the examples work online, but eventually I was forced to abandon it. It is just too badly put together to be worth ones time. A search online revealed that I was not alone in this impression.

The Unity programming in this project was primarily informed by Game Development with Unity (Menard, M 2012) using the terrain generation examples in Chapter 5 as a starting point.

3 Methods

In this chapter the approach to development of the game is described with specific information on the development environment and tools. A mood board for the game is presented along with a game design. A description of some design patterns used in the game coding is included along with the some of the techniques used to build a game level. Finally the user evaluation approach is discussed.

3.1 Development Method

This was a design and build project with evaluation both of the technical aspects of skiing game development using a head mounted display as well as a user evaluation of the delivered game. Using the Spiral model software design process (INM375 2012, lecture 2, slide 38) the project was built in a series of prototype stages in an iterative fashion, where each iteration was a milestone on the road to the final game experience. These milestones are documented in the Gantt chart in Appendix A and served to drive the development forward towards a timely completion.

3.2 Development Environment

At the start of the project, a decision had to be made regarding which development environment to use for the project. There are currently a number of ways to develop for the Oculus Rift. One is to use the Oculus Rift software development kit (SDK) library, which is compatible with Windows, Linux or Mac. Another option is to use a game engine which supports the Oculus Rift. As of the 4th July 2013 the game engines that have Oculus Rift integration are Unity3D v4 Pro, Unreal Developer's Kit (UDK) and fully licensed Unreal Engine 3 (UE3).

In version 0.2.3 of the Oculus Rift SDK, released on the 3rd July 2013, a C++ API library is provided along with sample code that uses DirectX rendering on Windows platforms and OpenGL rendering on Linux and Mac platforms. The project examples on Windows do not use OpenGL.

Investigating the supported game engines reveals that there are problems with the integration into UDK mainly because it doesn't support other middleware technologies such as ScaleForm. UE3 licensing is far beyond the cost range of this student. Unity integration seems to be well accepted and being used by a wide range of developers.

In light of the above factors, and taking into account the recommendations found in the literature review regarding the use of a game object component model, Unity was chosen for the prototype development environment. Unity allows three different scripting languages to be used. For this project C# was used so that its event/delegate model could be used to achieve a loosely coupled game architecture.

The development environment for the prototype game was

- Windows 7 Home Premium 32-bit SP1
- Unity Pro v4.1.5
- C#
- Home built PC (Intel E6750 CPU, 4GB DDR2 memory & 1GB AMD Radeon HD 5850)

If time allows, or the project demands it, then further development will be done using

- C++ in Visual Studio 2010 using OpenGL

3.3 Version control, backups and tools

I chose not to use a classic source control system such as RCS, CVS, SVN or Git for a couple of reasons: for one I am working alone and so don't need to track ownership of changes; and secondly because a project built within a game development environment such as Unity has dozens of critical settings which are embedded in the Unity project files. So I favour taking regular full project directory backups whenever I feel a significant piece of work has been done. I put these in a directory, along with other documentation and supporting files, that is protected by SugarSync (SugarSync Inc 2013) which is software that automatically synchronises files to cloud storage as well as to any additional machines you choose to share the directory with. This gives me protection against disk failure as well as flexible working across multiple machines.

I found that using cloud backup on the live project directory sometimes led to file locking conflicts when rapidly iterating through builds and test runs. This is because so many files are touched and updated in such a short time frame that the cloud backup would fall behind.

I have also found from past experience with Unity that it is possible for it to crash during development and also for the game to suddenly not work properly. It can be extremely daunting to try and resolve a problem like this, since it's not clear whether

the problem is in your code, or an errant UI selection in the Unity interface, or a bug in Unity. Having full working copies of the entire project is very comforting and useful.

I find Beyond Compare (Scooter Software 2013) to be a superb tool to perform file and folder side by side comparisons. This is very useful when reviewing all of the changes made between two versions of the project files. This more than compensates for not using a traditional source code management system.

3.4 Mood board & original concept drawing

A mood board helps to communicate a consistent vision of the game to the team creating it. It is a useful way to establish an art style, camera angles and suggest UI designs. For this project the following images serve to establish the look and feel guiding the design, in particular with a view to capturing the dynamic physical movement during play and the sense of immersion and involvement in the game.

Figure 4 - Mood board

Images used in the mood board (all accessed 7th July 2013):
http://fronttowardsgamer.com/wp-content/uploads/Motionsports_Skiing_bis2-265x300.jpg
http://static9.cdn.ubi.com/en-US/images/skitcm1910801.png
http://static1.businessinsider.com/image/51c378d2ecad04253a000009-1200/you-wear-the-rift-just-like-a-pair-of-ski-goggles.jpg
https://sphotos-a.xx.fbcdn.net/hphotos-frc1/s320x320/308204_243865542335377_804825564_n.jpg
http://download.xbox.com/content/images/66acd000-77fe-1000-9115-d8024d5389e6/1033/screenlg5.jpg
http://download.xbox.com/content/images/85cbff0d-f921-4d13-bb8a-40ba5dd0311d/1033/screenlg6.jpg
http://www.youtube.com/watch?v=HXbpa49W5qM (0:23)
http://d3nevzfk7ii3be.cloudfront.net/igi/DqQtJKlMnnJ2XjWV.huge

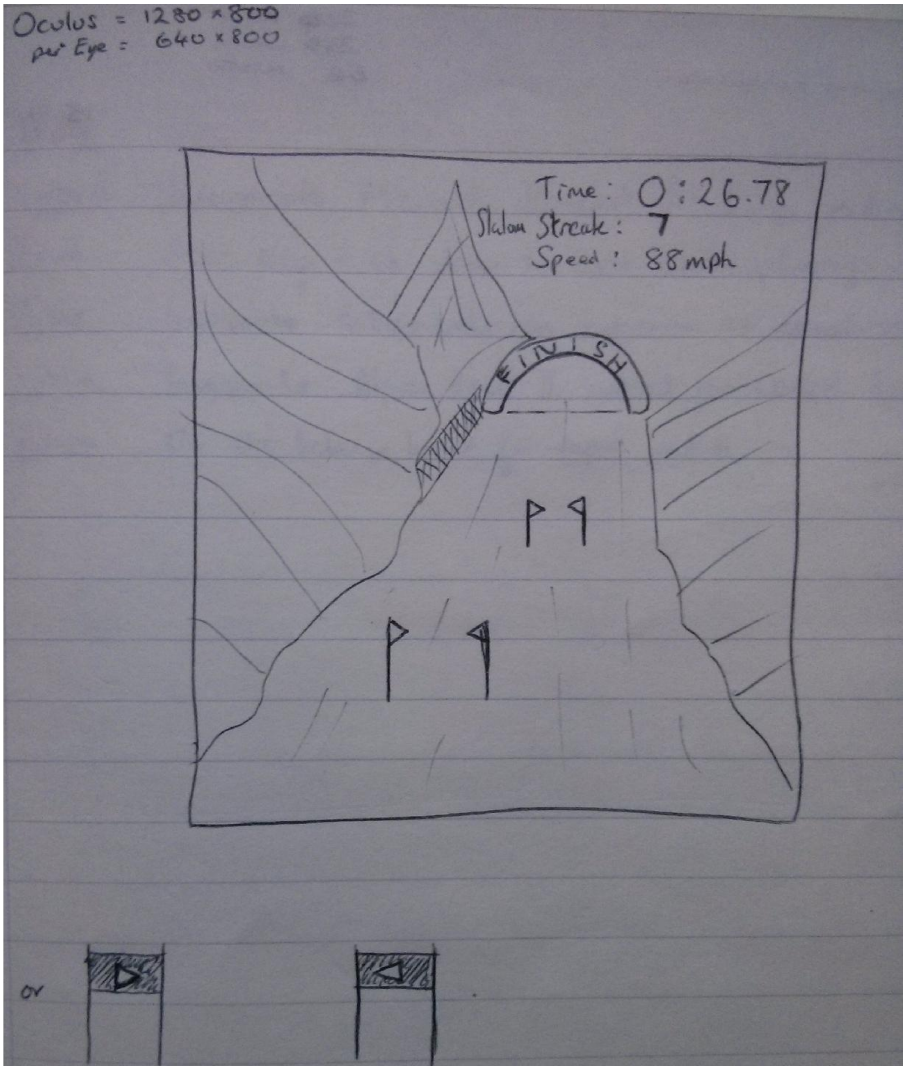


Figure 5 - Original concept drawing

Following on from the mood board the basic game look was sketched in a notebook as shown in Figure 5 this served to guide the game design.

3.5 Game Design

A full game design serves to communicate a consistent view of the game to all members of the team, however even a solo developer can benefit from going through the process of creating the game design documentation. Formalising the game design challenges the developer to think through all aspects of the game and how it should react.

3.5.1 Summary of the game

The objective of the game will be to traverse a slalom ski course. Movement will be controlled by the head motion detected when the player leans and crouches. Player profiles and a leader board will track progress and encourage competition between players based on time. Missing a gate will incur a time penalty. Reward icons will be displayed if a player beats target times and passes through every gate.

The edges of the racing area of the track will be coloured. The maximum speed that can be obtained will be higher within the racing lines than outside them.

The User Interface (UI) will indicate time since starting, current speed and gates passed through.

Beyond the scope of this project the game can be further expanded to include multiplayer racing, online leader boards, collectables, speed boost power ups and more mountain slopes.

3.5.2 Game play

Game play will primarily be a race against the clock, with time faults for missed gates. Leader boards will track progress between players.

3.5.3 Game progression

If the game is developed far enough to have multiple mountains to race on or multiple courses on a mountain, then progression will be in the form of unlocking further courses on completion of earlier ones.

3.5.4 Player controls

Once the player puts on the Oculus Rift headset then it is not appropriate to expect them to use the keyboard since they no longer have it in sight, and although a mouse could be used it would force the player to stand in a fixed position so she can operate the mouse, this breaks the immersion of being in the game world.

The design focus is to remove any reliance on keyboard and mouse to control the game once the headset is worn. A system is needed to enable the player to make

menu choices inside the game world. A 'look to select' menu system has been conceived. The menu choices will float in front of the player. The currently selected menu options are highlighted and further choices are made with a 'look to select' mechanism. When the player looks directly at a menu item a radial dial floats in front of the item and rotates from a sliver through to a full circle over the course of a couple of seconds, at which point the item is selected, and the associated action is taken.

Once the player has selected a course they are positioned at the top of the slope. On selecting 'Start Race' a three, two, one countdown appears and then skiing starts. Leaning from side to side is tracked via the tilt of the head to produce the slalom movement from gate to gate.

3.5.5 Difficulty system

There will be three difficulty levels, beginner, intermediate and advanced. Difficulty will be controlled via the level layout design. An easier level will have a lower top speed and gates that require less demanding turns to pass through. Higher difficulty levels will increase the maximum speed and require more demanding turns.

3.5.6 In-game UI

The game will have a very simple GUI layout. The time in minutes, seconds and fractions of a second will be displayed at the top of the screen. The top right will show the number of gates successfully passed, and the bottom left will show the current speed. The following diagram shows the GUI layout

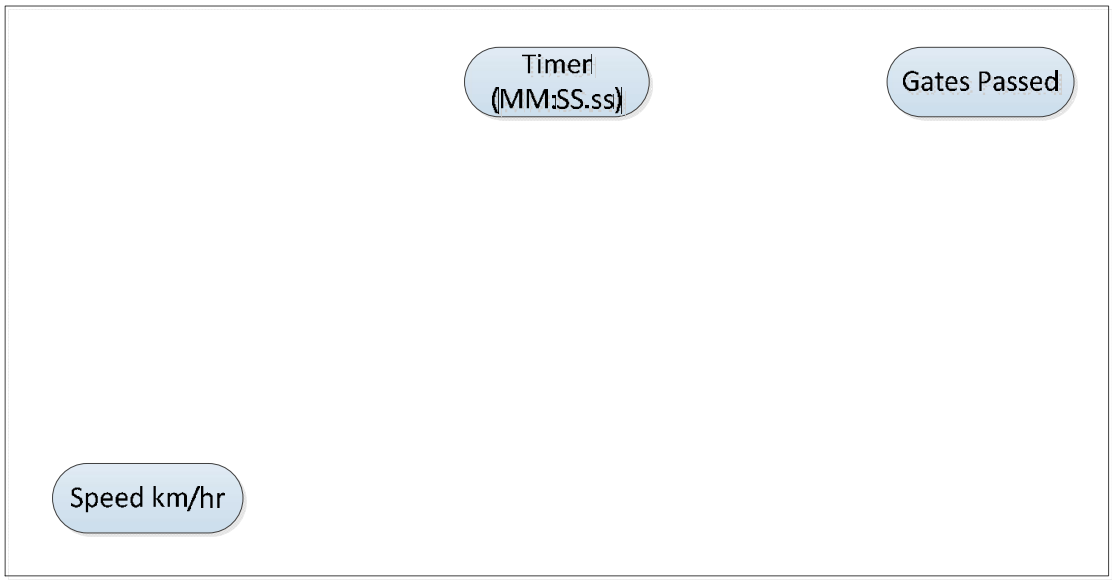


Figure 6 - Planned in-game GUI layout

3.5.7 Menu options and navigation

Logically the menu structure will look like this:-

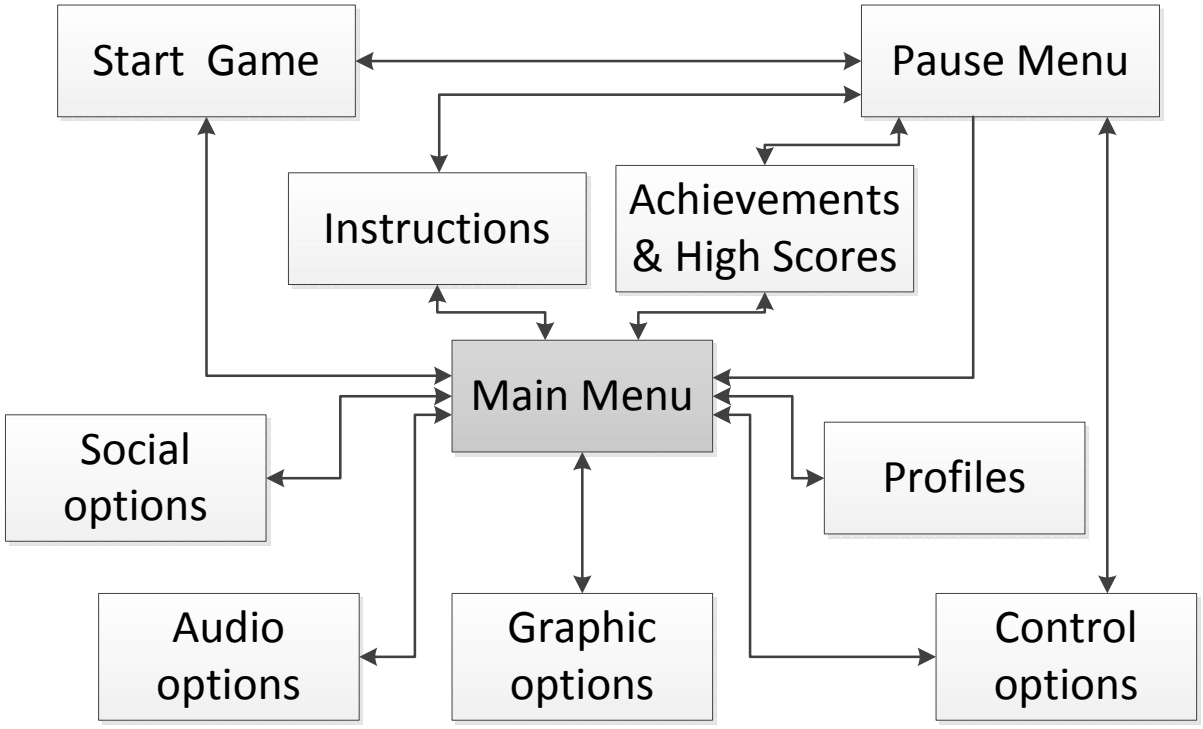


Figure 7 - Planned game menu structure

The remainder of the menu design documentation can be found in Appendix D.

3.5.8 Saving/Loading system

Each player will have a profile to which will be saved details of their achievements and levels unlocked as they progress. The profile will be updated automatically at the end of every run. Unlocking a new level will be rewarded with the relevant achievement.

Each level will have a name. From the game start screen the player will be able to choose the starting level. Initially only the first level will be displayed along with its name, the rest will show as question marks. As each level is reached it will then be available as a starting location.

3.5.9 Interaction matrix

One of the formal approaches to documenting a game design (Rollings & Morris 2004, p.494) is the creation of a token interaction matrix. This triangular format table documents the game logic that should be applied for a collision between each type of game object. It is a useful tool that encourages further contemplation of the overall game structure.

	Player	Second Player*	Slalom Gate Pole	Slalom Gate	Ski Boundary	Tree	Power-up	Coin
Player	X							
Second Player*	Collision, push player	X						
Slalom Gate Pole	Collision, slow down[1]	X / Slow down	X					
Slalom Gate	Score event	X / Score event	X	X				
Ski Boundary	Collision, slow down[1]	X / Slow down	X	X	X			
Tree*	Collision, slow down[1]	X / Slow down	X	X	X	X		
Power-up*	Pickup event	X / pickup	X	X	X	X	X	
Coin*	Pickup event	X / pickup	X	X	X	X	X	X

1 - A stumble animation was considered but discarded as too disorientating for a player wearing the Oculus Rift

* - Not planned to be implemented during this project

Figure 8 - Interaction Matrix

3.5.10 Collision detection

Collision detection in this game will be based on fast and efficient intersection testing which will be sufficiently accurate to be fair. For a player to gain credit for passing through a gate both of their feet must pass between the gate poles. This is similar to official Giant Slalom rules which also require the tips of the skis to pass between the gate poles.

3.5.11 Collision response

Response to collisions with ski gate poles will only result in a slight loss of speed, collisions with boundaries and trees will result in more significant speed loss and where sensible course correction back onto the track. A direct impact with a tree will lead to a wipe-out, however the impact of the visualisation of this will be assessed before it is included, since it may be too disorientating for a player wearing an Oculus Rift, particularly if they are standing to play and don't have any support to hold on to.

3.5.12 Scoring

To remain true to the racing sport being simulated scoring will simply be the time taken from the top to the bottom of the course with time penalties for missed gates.

3.6 Design Patterns

Design patterns are templates for reusable code. They aim to inform best practices for object orientated software engineering. An excellent book on Design Patterns is "Head First Design Patterns" (Freeman et al. 2004). Some of the design patterns used in the project are described below.

3.6.1 Singleton & Factory Design Patterns

It was particularly important to use the Singleton design pattern with regards to the main player GameObject in Unity. Unity's default approach to loading a new level is to destroy all of the current GameObjects and instantiate all of the GameObjects associated with the new level. This may be acceptable for some games, however when working with the Oculus Rift it is crucial to maintain a consistent frame of reference to the player's head position. In Unity the Oculus Rift camera is attached to the player's GameObject. When it is initialised the current position of the Oculus Rift is taken to be the neutral position, i.e. the player looking straight forward. This caused a problem when loading new levels by with the 'look to select' menu, since the new camera would be initialised with the player looking off to the side at the menu option tile.

After some investigation and trying various workarounds it was found that it is possible in scripting to tell Unity not to destroy a GameObject on level load.

Unfortunately this triggered a new problem when a new level was loaded. For a brief moment there would be two concurrent instances of the Player GameObject whilst the Singleton code was running to determine that the second one should exit since the first one already existed. This caused the graphics rendering to stop tracking the player's head position.

This problem was solved by creating a GameManager GameObject which effectively implemented a Factory Design pattern approach and also adhered to the Singleton principle. In every game scene in the Unity Editor the Player GameObject was removed and replaced by the GameManager GameObject. GameManager is responsible for ensuring that a Player GameObject is created if one does not exist.

3.6.2 State Pattern (Finite State Machine)

The Player GameObject maintains state for the player. The following diagram shows the various states used in the game. When the Player GameObject is first created it is initialised into the 'Ready to Ski' state and then OpenMenu() is called to bring up the menu interface.

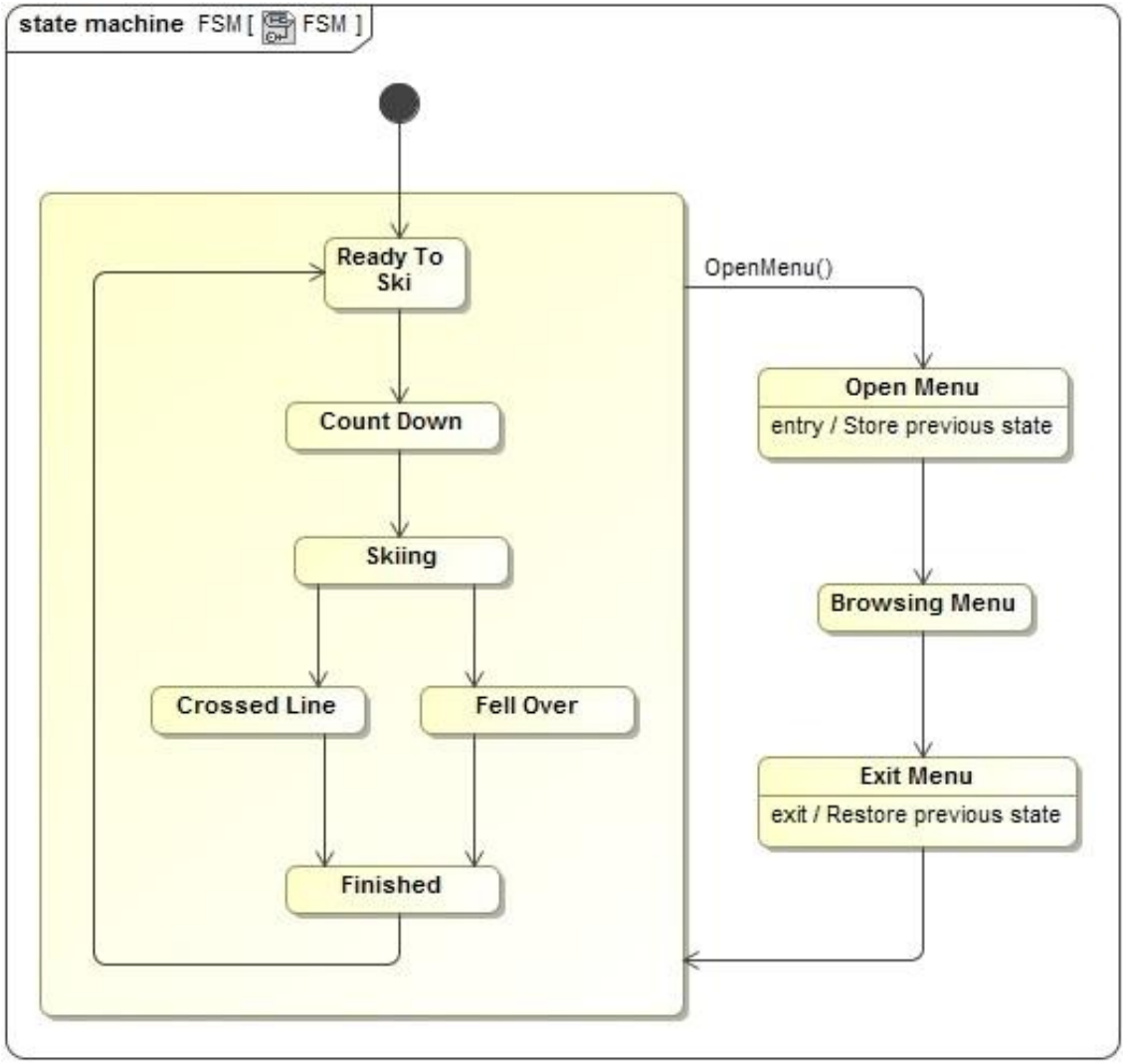


Figure 9 - Player finite state machine

The main purpose of each state is as follows

State	Description
Ready to ski	Player is on starting line ready to race
Count down	Player has selected 'Start Race' in menu and is now being counted down to start, 3..2..1..GO
Skiing	Player is in control and skiing
Crossed line	Player has crossed the line and is brought to a halt. Their final score is calculated and a high score is added if appropriate
Fell over	Player has fallen over so the level is stopped early
Finished	Player has come to a halt, call OpenMenu()
Open menu	This state can be triggered at any time. It causes the menu items to fly into position to give the player 'look to select' control over the game.
Browsing menu	The player has control, the floating menu is fully opened
Exit menu	The menu items fly away. If a menu option has just been selected then its associated action is triggered. This is either a level load or a 'Start Race' which transitions the player through 'Ready to Ski' to 'Count down'

3.6.3 Observer Pattern (Publisher-Subscriber)

The game will be built around an event driven architecture to separate out key classes and responsibilities. Scripting within Unity is automatically incorporated into the core game loop such that at predefined stages within the main game loop certain functions, if present, will be called in each active script attached to a gameObject. For instance on gameObject instantiation Awake() is called before Start(), then once the game is up and running Update() is called once per frame. Physics based calculations should be placed in FixedUpdate() which can be called more than once per frame. Beyond this there are many functions that will be called if an event is triggered within Unity, such as OnCollisionEnter(), which is called if the physics engine detects this gameObject has collided with another.

Beyond the Unity framework C# scripting was chosen with a view to using its event delegate model to build a flexible game architecture that reacts to events generated during gameplay. The *Unity 4.x Cookbook* (Smith & Queiroz 2013) states on page 322 "Delegates and events implement the publish-subscribe(pubsub) design pattern" and

goes on to explain that this form of “loose coupling” separates out the coding, and maintenance, of the event generation code (the publisher) from any number of event consumers (the subscribers) which is a desirable feature of object-oriented code.

For instance when the playerStatus object registers a change in state, as shown in the Finite State Machine diagram in Figure 9 of section 3.6.2, then any other gameObject can choose to react to that change in state.

An example of this occurs with the floating menuItem gameObjects each of which subscribes to the menuItemsActivationEvent which is signalled when the menu is opened or closed. Each menuItem then transitions from invisibly far away into position around the player, or zooms away.

3.7 Terrain acquisition

In order to increase appeal to real skiers it was decided to pull the real terrain heightmap data from Google Earth via Google Sketchup Pro (now owned by Trimble). This data source is available for personal non-commercial use as described on Googles website (Google 2013).

Instructions on how to do this were found on the web (Alpenglow Games 2013). This data source does not include any of the textures seen when viewing the terrain in Google Earth, so adding trees and finding a route across the terrain was done by looking at ski resort route maps and attempting to follow a similar route. Some of the terrain data included very rough sections which caused the first person view to bounce in a disconcerting fashion, so the Unity terrain smoothing tool was used to improve the smoothness of movement.

The following pages show the main steps taken from finding a mountainous terrain in Google Earth through to a level in the game.

3.7.1 Find terrain in SketchUp Pro

Having opened SketchUp Pro click the 'Add Location' in the Google toolbar, type in the name of a ski resort and click the Search button. Positioned the map to capture the desired landscape then click the 'Select Region' button, followed by the 'Grab' button.

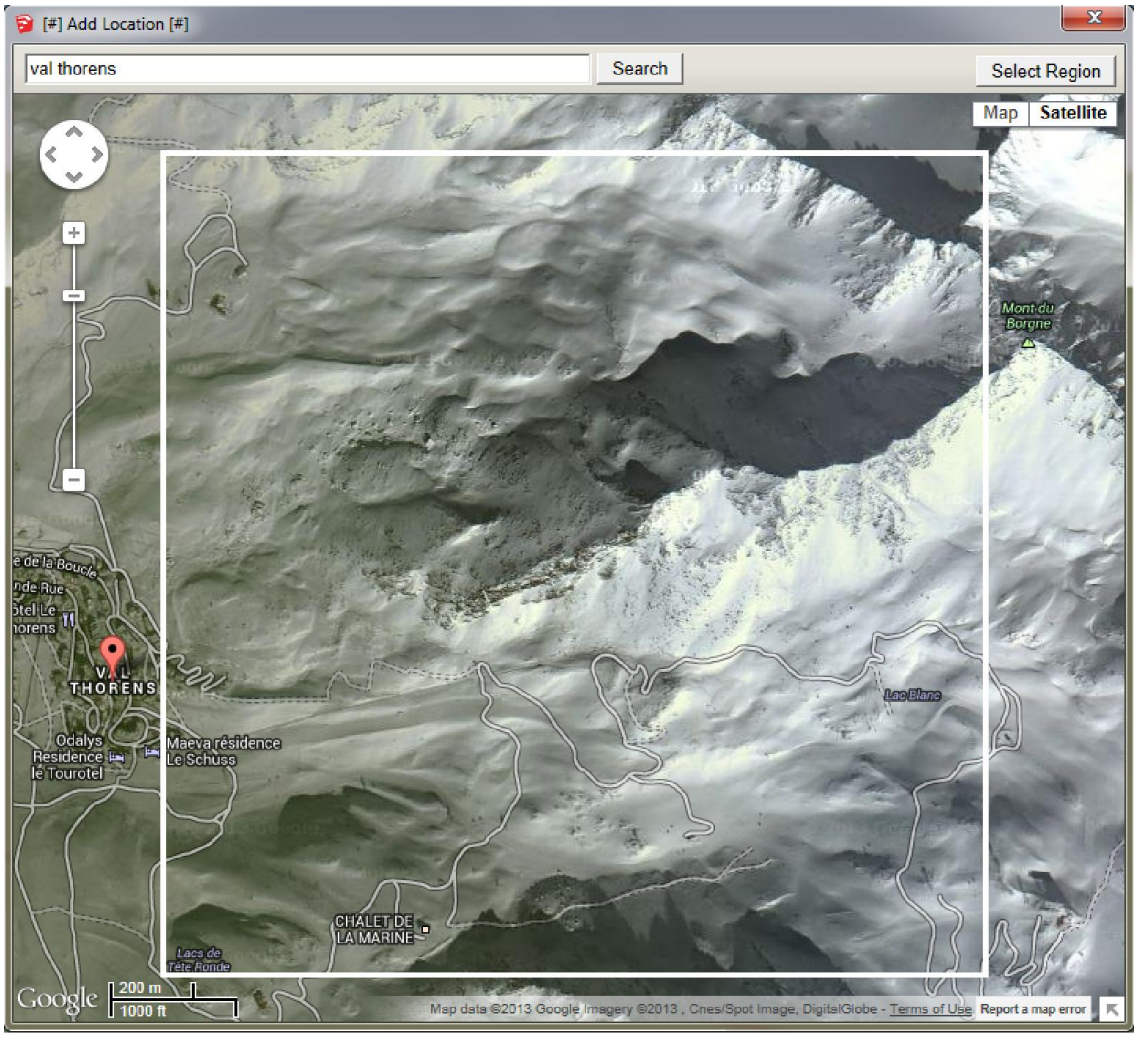


Figure 10 - Google Earth imagery in SketchUp Pro

3.7.2 Viewing Terrain in SketchUp Pro

At this point the terrain appears as a flat quad. Clicking the 'Toggle Terrain' button applies the heightmap data to the textured quad and shows the terrain layout clearly. This is now exported as a .fbx file.

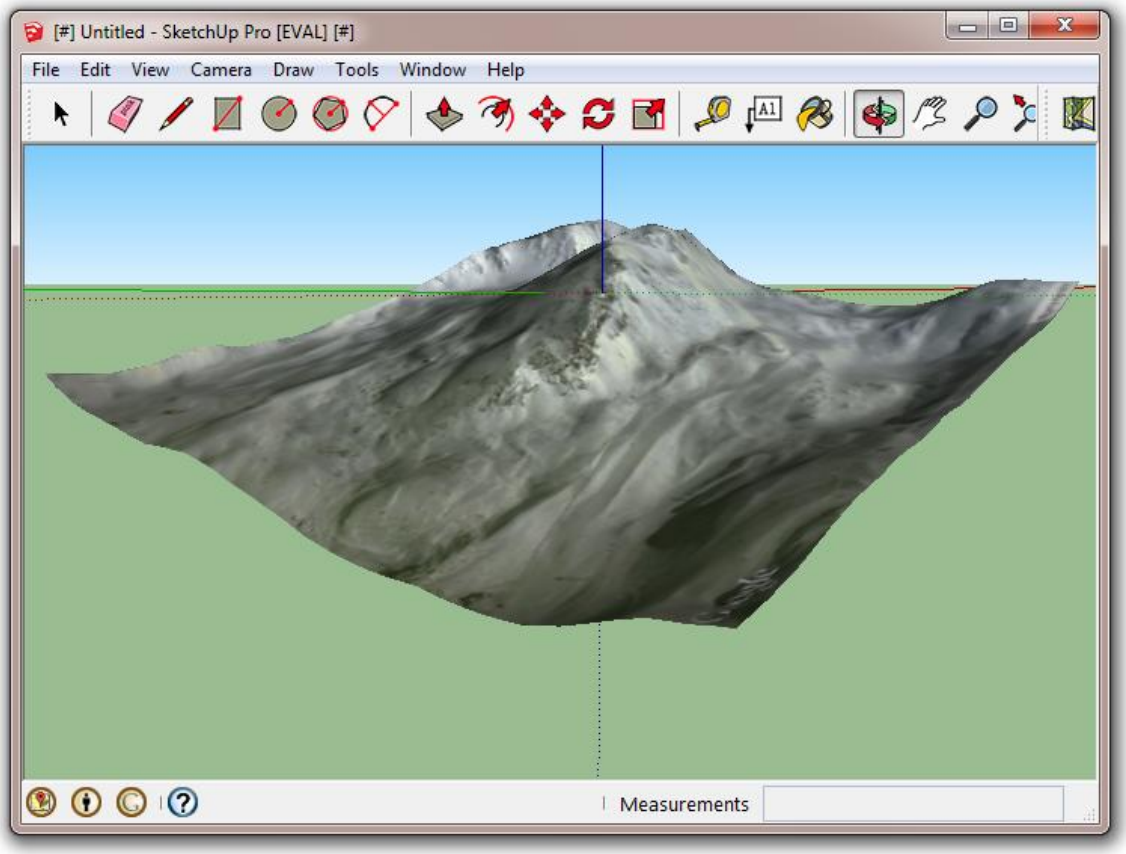


Figure 11 - Google Earth Terrain in SketchUp Pro

3.7.3 Unity Terrain

The .fbx file is imported into Unity as an asset. Unity shows the mesh shape of the object. A standard flat terrain is then created and the imported asset, containing the heightmap mesh, is dragged onto the new terrain as a child object. The Unity flat terrain and the imported mesh object can both be seen in the following picture.

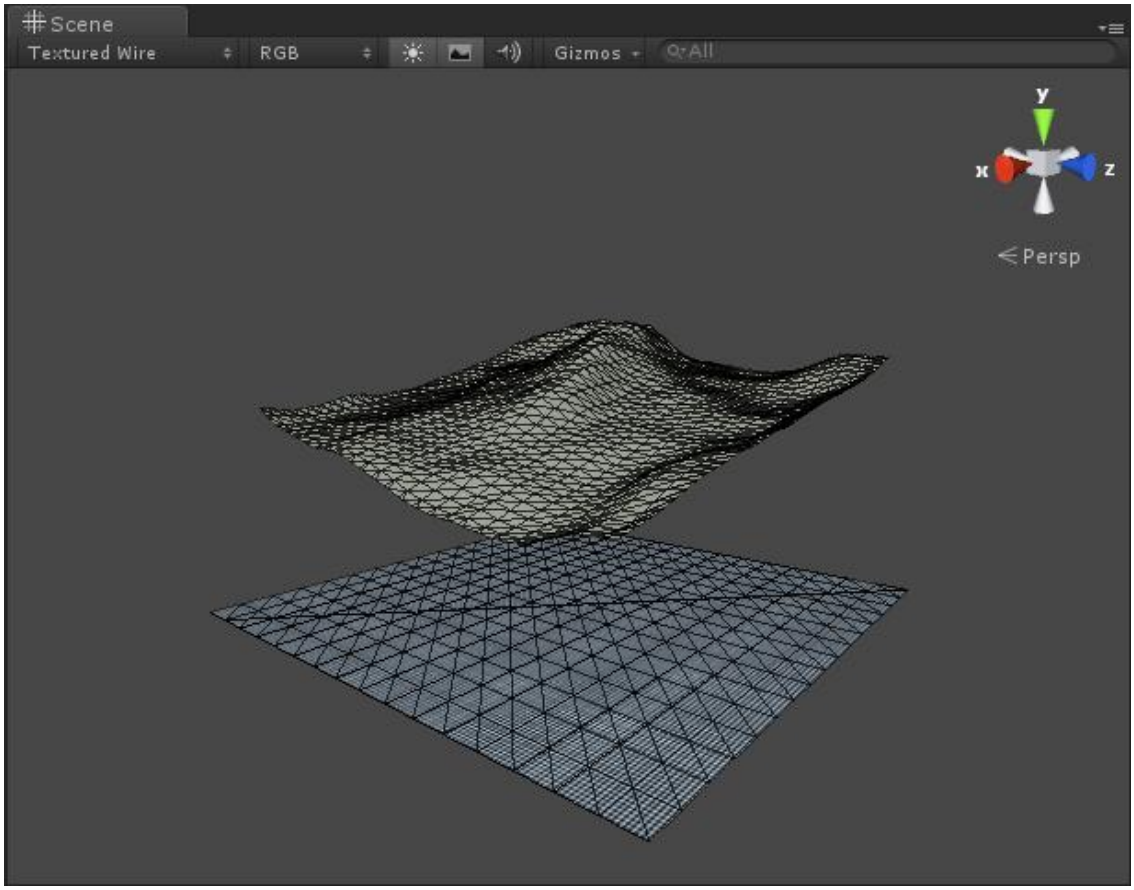


Figure 12 - Terrain heightmap mesh in Unity

3.7.4 Unity Terrain mesh

The object2Terrain script is then invoked from the Unity menu which applies the imported mesh shape to the Unity Terrain object.

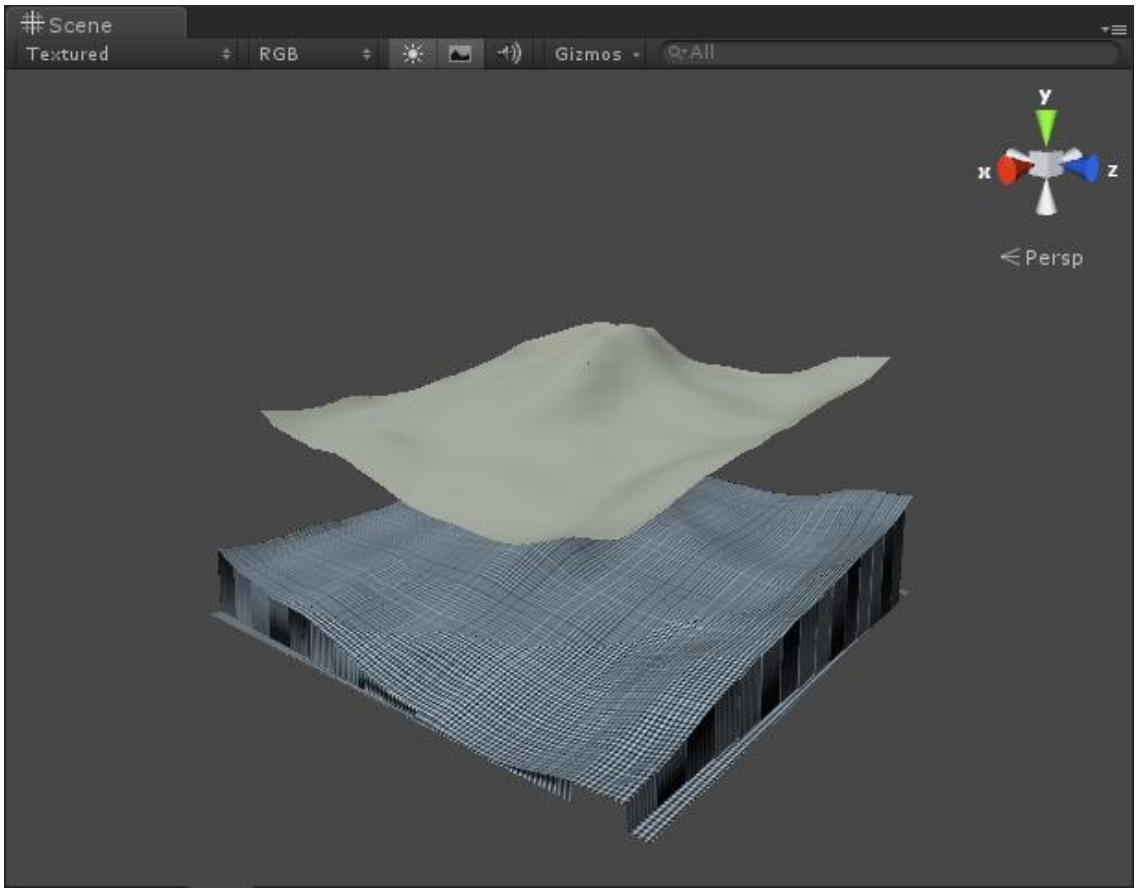


Figure 13 - Unity Terrain with heightmap applied

3.7.5 Terrain populated

At this point the imported asset can be deleted from the Unity terrain object. Only the heightmap data has made it through this process from Google Earth to Unity Terrain. The textures are not available, so now terrain features such as ground texture and trees are added manually using the Unity terrain tools.

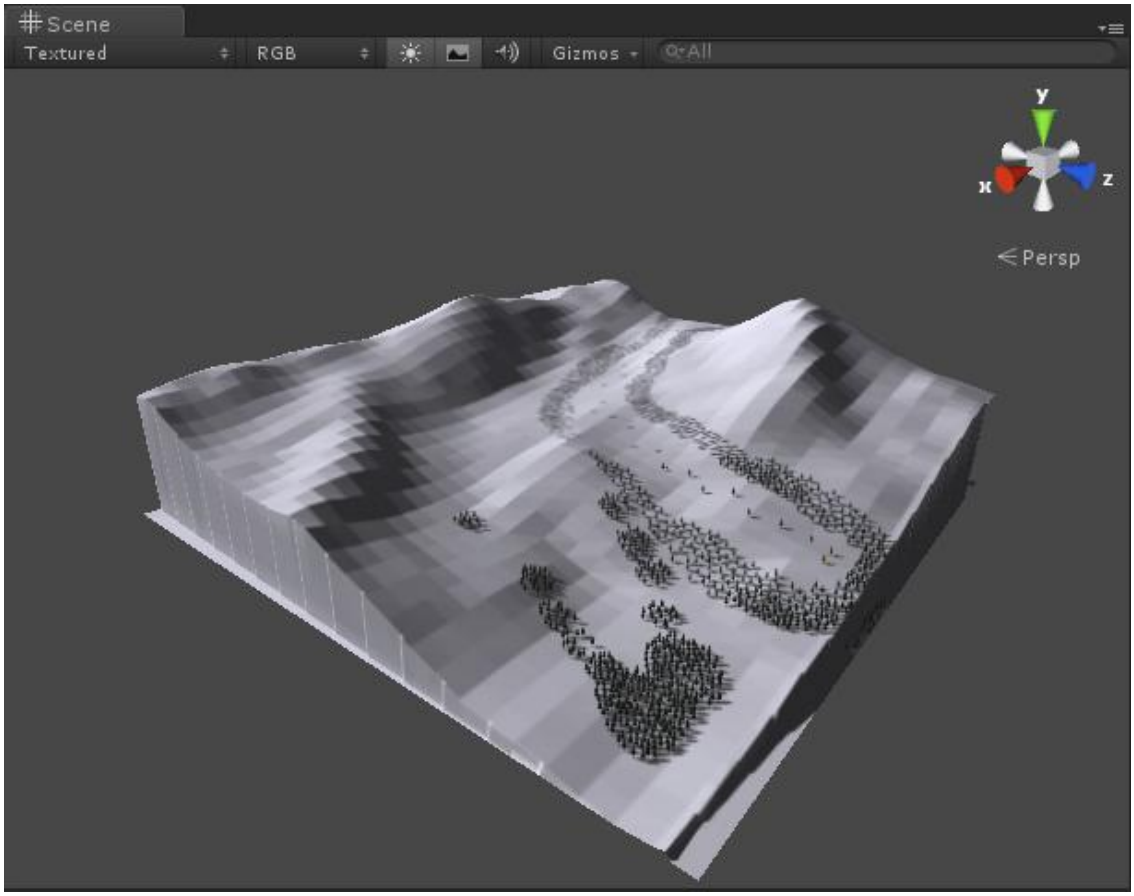


Figure 14 - Unity terrain with trees

3.7.6 Course creation

Course elements such as the player start position, slalom gates and the finish line are placed on the terrain. For the beginner levels a single tree was positioned next to each gate to give the player maximum visibility of where the next gate would be, this was especially important because of the low resolution screen within the Oculus Rift. The slalom gate poles and flags were also made much thicker to increase their visibility.

Figure 15 - Unity course design

3.7.7 Final view

A view of the end of the course can be seen below.

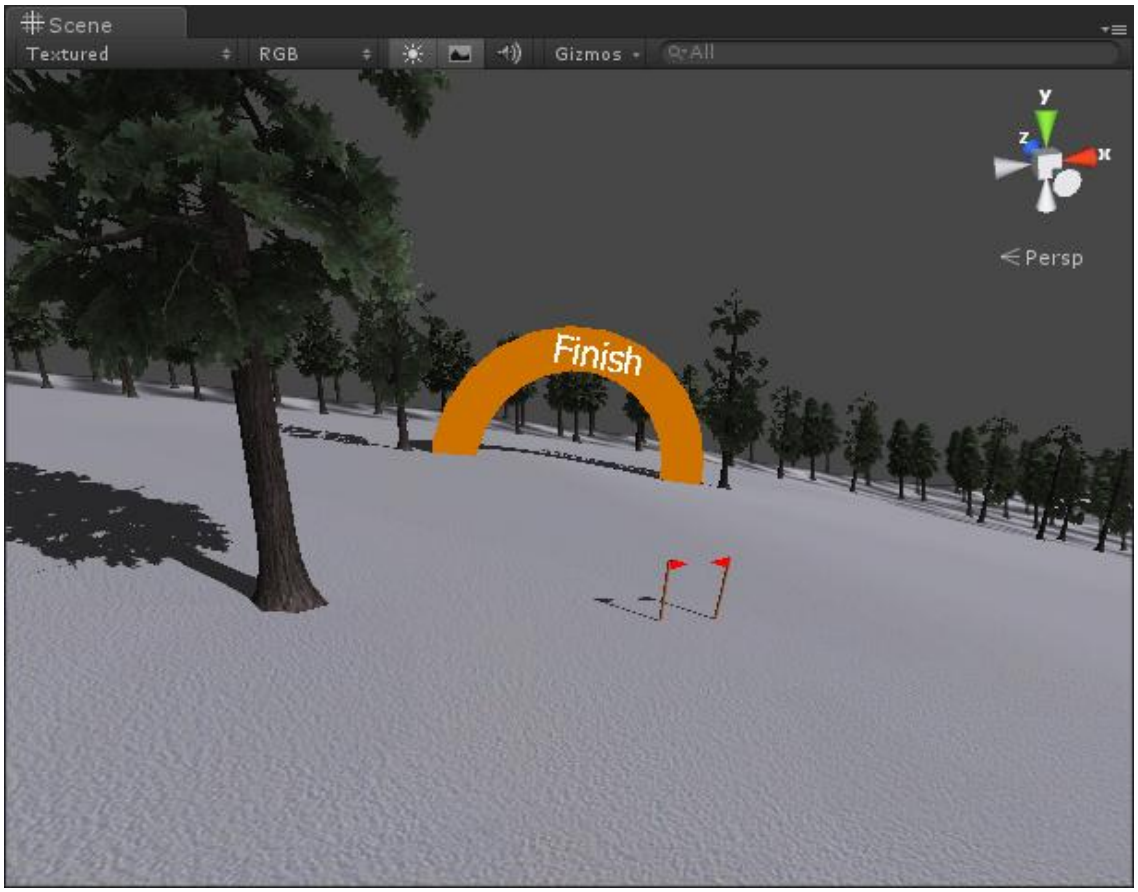


Figure 16 - Player view of course finish line

3.8 Testing

During development an 'it always runs' approach was taken. Each feature, as it was added, was fully tested and the game could always be played to the limit of its currently implemented features. This approach helped to identify problems early and also provided motivation since progress could always be seen.

Testing of code such as the high score table insertion routine was done by calling the routine with a series of test values and observing the result in each case. Unity is supplied with an integrated code editor and debugger called MonoDevelop. MonoDevelop can be instructed to attach to a process, in this case the Unity Editor, and break points can be set in the code. When the game is run and the break point is hit then control is given to MonoDevelop which enables the coder to view variable values and step through the code.

Testing of the InsertScore(float seconds) routine in HighScoreController.cs was done by making a series of calls to InsertScore() and monitoring the code flow and variable values. For example starting with a freshly initialised high score table where each entry is 540 seconds, the following tests were performed :-

Test	Expected Result
InsertScore(90.0f)	New score placed in 1st position and all lower scores shifted down one place, last score overwritten.
InsertScore(100.0f)	New score placed in 2nd position and all lower scores shifted down one place, last score overwritten.
InsertScore(88.0f)	New score placed in 1st position and all lower scores shifted down one place, last score overwritten.
InsertScore(600.0f)	No change to high score table.
InsertScore(540.0f)	No change to high score table.
InsertScore(88.0f)	New score placed in 2nd position and all lower scores shifted down one place, last score overwritten.

3.9 User study

The following paragraphs describe the preparation and execution of the user evaluation study.

3.9.1 Practise evaluation

A practise user evaluation was performed at a social gathering with friends, a dozen of who tried the game. A number of lessons were learnt from this regarding how to perform the evaluation. In particular based on the test group it was decided not to have a chair available for participants and not to perform the Oculus Rift configuration setup for each person. It was observed that even a slight offset from the optimal headset position led to out of focus areas of the display, which proved to be much more noticeable than the visual difference after running through the configuration utility. It is assumed that for longer periods of use it would be advisable to spend the 4-5 minutes setting up a configuration profile for each user.

3.9.2 Real evaluation

The real user evaluation, the results of which are in this project report, was performed all day at the Snow & Rock store in Chertsey on Saturday the 16th November. Members of the general public, who visited the store to buy ski gear and have boots fitted, were invited to try the slalom skiing game using the Oculus Rift headset.

When a shopper showed an interest in the game they were given a brief explanation of the game and the headset. They were shown how the physical movement of the headset changed the view and given a demonstration of the 'look to select' menu system.

They were told that steering was accomplished by tilting the head using the phrase 'by moving ear to shoulder' to ensure they had a clear idea of how the game was controlled.

They were then helped to put on the headset, adjust it for comfort, and then given a pair of ski poles to hold which ensured they had a way to stabilise themselves if they lost balance.

After they finished a run or two of the slalom course they were then asked if they would be willing to fill in the single sided questionnaire as described in Section 3.9.4.

Each questionnaire was filled in anonymously, and without oversight, to encourage honesty.

3.9.3 Recording the results

After the evaluation day each sheet was numbered and a spreadsheet was created to hold all of the responses received, a snippet of which is shown in Figure 17.

	A	B	C	D	E	F
1	Question#	1	2	3	4	5
2	Sheet #	Sex	Age	Ski Rating	Experienced Games	Rift Comfortable?
3	1	Female	18-30	Intermediate	Yes	Mildly uncomfortable
4	2	Male	18-30	Intermediate	Yes	Comfortable
5	3	Male	50+	Advanced	No	Comfortable
6	4	Male	30-50	Advanced	No	Mildly uncomfortable
7	5	Male	18-30	Intermediate	Yes	Mildly uncomfortable
8	6	Male	50+	Advanced	No	Comfortable
9	7	Female	30-50	Intermediate	No	Comfortable
10	8	Female	50+	Intermediate	No	Comfortable
11	9	Female	50+	Intermediate	No	Comfortable

Figure 17 - User evaluation responses spreadsheet

A summary sheet was created which counted up all of the responses and showed the total number of responses for each question, this was to verify that the expected number of answers were present. See Figure 18.

	B	C	D	E	F
4					Total
5	Which age range are you in?	18-30	30-50	50+	
6		9	17	6	32
7		28%	53%	19%	

Figure 18 - User evaluation summary spreadsheet

Formulas were used in the summary sheet rather than hand typed results, again to ensure accuracy in the results, as shown in Figure 19.

	B	C	D	E	F
4					Total
5	Which age range are you in?	18-30	30-50	50+	
6		=COUNTIF(Answers!$C$3:$C$34,C5)	...	=COUNTIF(Answers!$C$3:$C$34,E5)	=SUM(C6:E6)
7		=C6/SUM($C6:$E6)	...	=E6/SUM($C6:$E6)	

Figure 19 - User evaluation summary spreadsheet formulas

Verification columns were added to count the number of answers provided for each question against the number of matching answers counted for each question. Errors were temporarily introduced to verify the formulas and validation cells were working correctly.

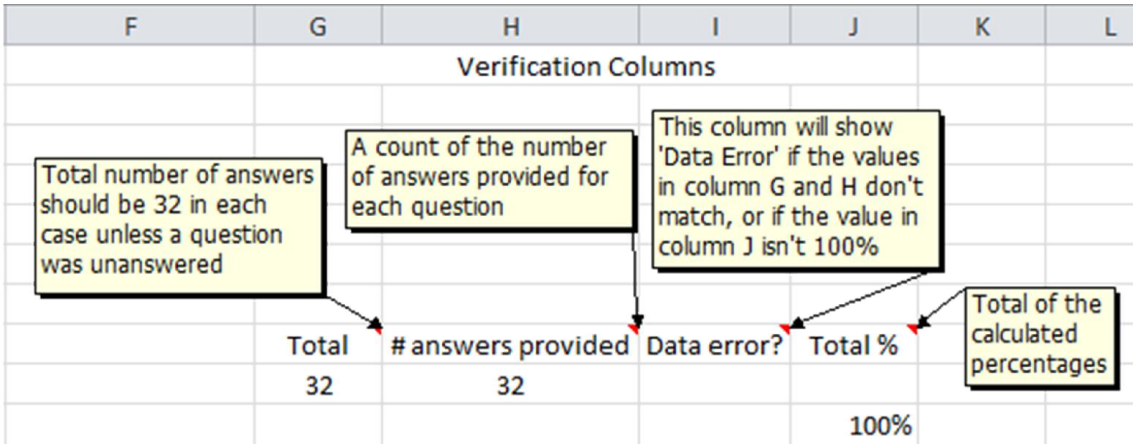


Figure 20 - User evaluation spreadsheet verification cells

The count of the number of answers provided in column H was calculated using excel to count the size of the data set and subtract the number of blank cells, e.g.

=ROWS(B$3:B34)-COUNTBLANK(B$3:B34)

The results of the evaluation are discussed in section 3.9.5.

3.9.4 Evaluation form

The following feedback form was designed in order to answer the research questions that were informed by the literature review. The form sought to find out how immersive and fun the experience was and whether it generated any sense of disorientation or motion sickness.

As described in Section 3.9.2 members of the general public were invited to try the game for a few minutes and then confidentially fill in the feedback form shown on the next page.

Virtual Reality Slalom Skiing Game Evaluation Questionnaire

Please ring your choices.

If you want to cancel a ringed choice draw a cross on top.

1. Are you male or female?	Male	Female	
2. Which age range are you in?	18-30	30-50	50+
3. How would you rate yourself as a skier or snowboarder?	Beginner	Intermediate	Advanced
4. Are you an experienced games player?	Yes	No	
5. Did you find the Oculus Rift (head mounted display) comfortable to wear?	Uncomfortable	Mildly uncomfortable	Comfortable
6. When you looked around did the view track accurately to your head movement?	Yes	No	
7. Did you experience any sense of disorientation or motion sickness?	Yes	No	
If yes above please provide additional comments			
8. How would you rate the experience?	Good fun	Fun	Not fun
9. Did you sit or stand?	Sat	Stood	
10. Did you start to move from side to side similar to slalom movement?	Yes	No	
11. Did you put weight on to your turning foot?	Yes	No	
12. Did you find the experience immersive?	Yes	No	
13. Did you find the 'look to select' menu control intuitive and easy to use?	Yes	No	
14. How well did your physical steering movements match the game movement?	Matched well	Matched sometimes	Did not match well
15. Do you think this game would help someone who is scared of heights?	Yes	No	
16. Do you think this game would help someone who is new to skiing or snowboarding?	Yes	No	

17. Did this experience make your visit to Snow & Rock more enjoyable?	Yes	No

Any other comments?

Figure 21 - User evaluation questionnaire

3.9.5 Feedback from the general public

A total of 32 responses were collected after spending a day at Snow & Rock on Saturday the 16th November 2013. The manager of the store extended an invitation to return to the store to participate in a special promotional evening on the 5th December 2013. The invitation was accepted but unfortunately due to bad traffic no members of the public came into the store that evening and no additional evaluations were completed.

The feedback form is shown below in Figure 22 with the percentage number of times each choice was made is shown in bold.

1. Are you male or female?	Male **66%**	Female **34%**	
2. Which age range are you in?	18-30 **28%**	30-50 **53%**	50+ **19%**
3. How would you rate yourself as a skier or snowboarder?	Beginner **28%**	Intermediate **50%**	Advanced **22%**
4. Are you an experienced games player?	Yes **41%**	No **59%**	
5. Did you find the Oculus Rift (head mounted display) comfortable to wear?	Uncomfortable **0%**	Mildly uncomfortable **22%**	Comfortable **78%**
6. When you looked around did the view track accurately to your head movement?	Yes **97%**	No **3%**	
7. Did you experience any sense of disorientation or motion sickness?	Yes **50%**	No **50%**	
If yes above please provide additional comments			
8. How would you rate the experience?	Good fun **68%**	Fun **29%**	Not fun **3%**
9. Did you sit or stand?	Sat **9%**	Stood **91%**	
10. Did you start to move from side to side similar to slalom movement?	Yes **66%**	No **34%**	
11. Did you put weight on to your turning foot?	Yes **44%**	No **56%**	
12. Did you find the experience immersive?	Yes **93%**	No **7%**	
13. Did you find the 'look to select' menu control intuitive and easy to use?	Yes **97%**	No **3%**	
14. How well did your physical steering movements match the game movement?	Matched well **47%**	Matched sometimes **38%**	Did not match well **16%**
15. Do you think this game would help someone who is scared of heights?	Yes **48%**	No **52%**	
16. Do you think this game would help someone who is new to skiing or snowboarding?	Yes **68%**	No **32%**	
17. Did this experience make your visit to Snow & Rock more enjoyable?	Yes **100%**	No **0%**	

Figure 22 - Feedback results

These are the comments that accompanied answers to question 7 (Did you experience any sense of disorientation or motion sickness?)

- “A little like I was going to fall over”
- “I am susceptible to spinning sickness and I had this sensation”
- “Glasses wearer. Extremely realistic for slalom skiing turn early”
- “At the end felt bit sick”
- “Felt dizzy” [note, this person answered ‘No’ to the question]
- “Only slightly”
- “More unsteady when I came off machine”
- “When hitting the flag & the finish”
- “Not in a negative just the experience of not seeing”
- “Just felt weird”
- “Slight disorientation”
- “Strange moving without your legs moving”
- “Felt slightly similar to car sickness”

These are the comments found in the final box

- “Very immersive. Can see progression would come with use”
- “Stand on skis”
- “Waving head about”
- “Good fun!!”
- “I found it challenging as I was taught to keep my head fixed for so many years”
- “Good level of mental immersion but the controls are too easy just to operate with your neck”
- “Interesting technology - maybe not quite ready for consumer use but good fun”
- “Thanks”
- “Thank you. Very strange feeling!”
- “Good luck, great idea, when getting lost needs a prompt, would be good to see your ski tips”
- “Cool stuff, really interesting gadget”
- “Very strange experience, but good fun”

Overall the results were very encouraging. They can be broadly summarised by the following statements.

- Everyone said it made their visit to Snow & Rock more enjoyable
- Almost everyone had fun (1 person didn't)
- ~80% of people found the Oculus Rift to be comfortable the others reporting it to be mildly uncomfortable, no one said it was uncomfortable
- Half of the people reported some disorientation or motion sickness, in some cases quite slight
- Over 90% of people found the experience to be immersive, with two thirds reporting that they moved in a similar fashion to slalom skiing, almost half reporting that they found themselves shifting their weight to their turning foot.
- Over 95% of people found the 'look to select' menu system intuitive and easy to use
- ~50% of people found the steering movement matched well to their movement, 16% reporting it didn't match well at all.

From this feedback the most important areas of the game that need further work are the steering movement and the sense of disorientation that was experienced. The tilting of the head to steer does not accurately match the motion that a slalom skier performs in real life. In the game the amount of head tilt required is quite small & subtle whilst the player stays on the racing line, however if rapid course correction is needed then the amount of head tilt required becomes unrealistic. It might be possible to solve this by applying a non-linear sideways force based on the length of time that a turn has been initiated; however such an approach will need very careful tuning to avoid it becoming erratic and unintuitive for the player.

Disorientation within the Oculus Rift bares more extensive research with players spending a longer period of time evaluating it. It is this author's opinion that some of the disorientation experienced will reduce as familiarity with the movement controls increases and through repeated use of the Rift. In this evaluation players rarely spent more than 2 minutes playing the game, those that spent longer found movement control became much easier as they learnt the sensitivity of the system.

4 Results

This chapter describes the results of the project with a particular focus on how the original objectives were met. Images of the game in action are shown along with explanations of some of the key systems, in particular the unique and novel 'look to select' menu system. In addition some of the challenges faced are described in detail.

4.1 Images of the game

The following pictures were taken of the game being played. The computer monitor displays the images that are being displayed in front of each eye within the Oculus Rift. The person wearing the Rift sees one seamless image of the world around them and no black borders. See section 4.7 for further discussion on this.

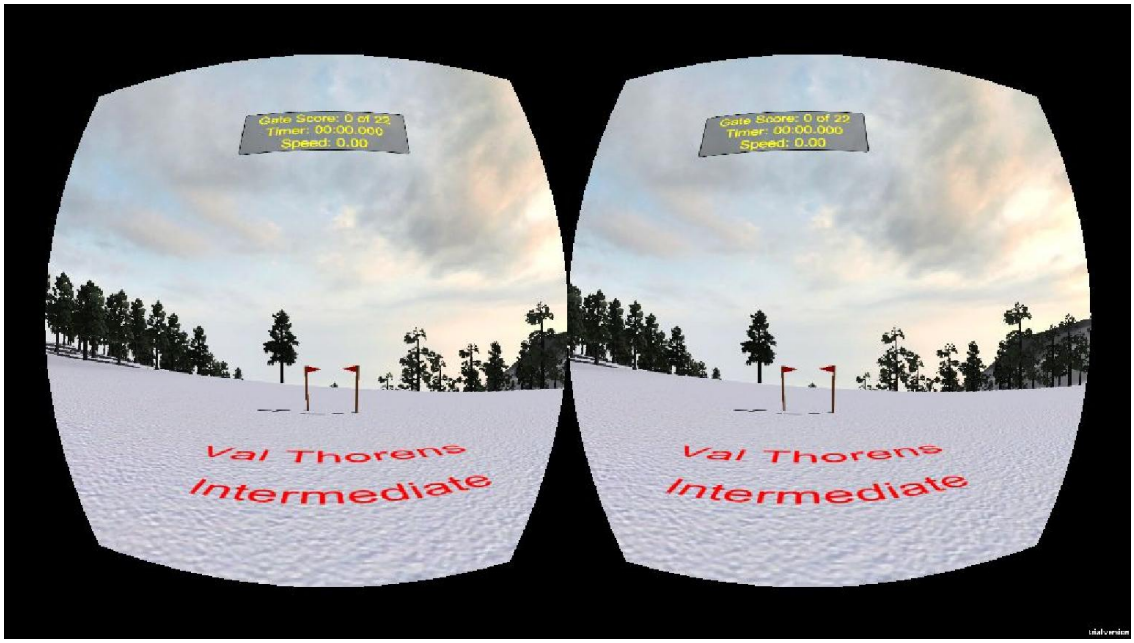


Figure 23 - View from start line

Figure 23 shows the player's view of the start of the Val Thorens course. The course name and difficulty are written into the snow in front of the player, real-time shadows are created for nearer objects, such as the flag poles, with baked in lighting used on the distant trees, this is demonstrated in Figure 24, and in Figure 25 the renderer has blended out the baked tree shadow and replaced it with a high detail real-time shadow, this technique is discussed in section 4.10.

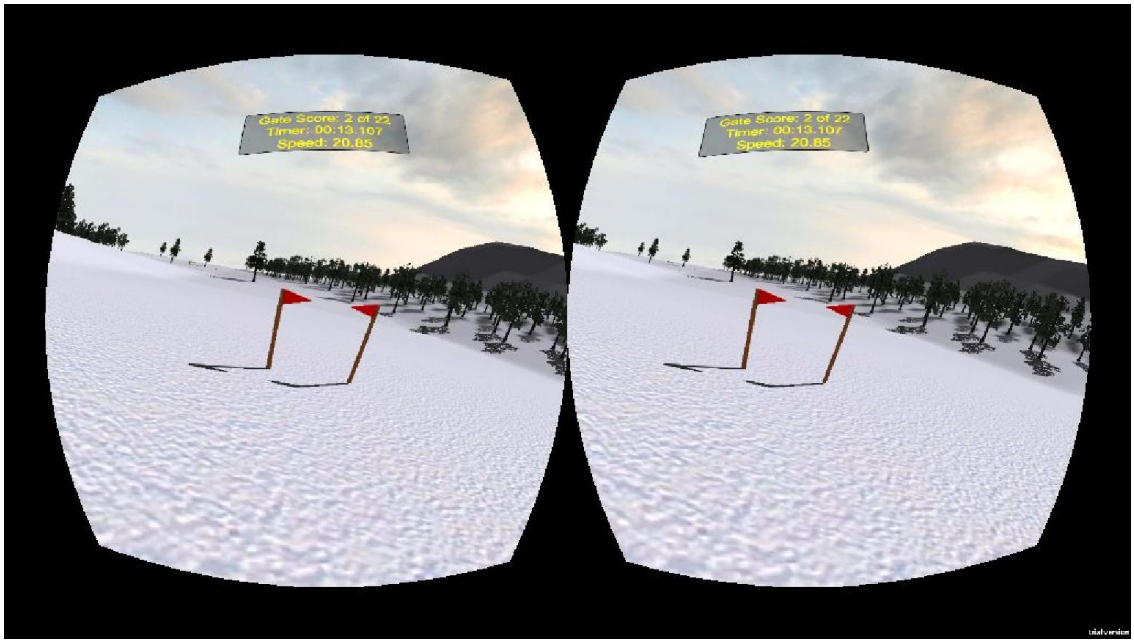


Figure 24 - Skiing, approaching a slalom gate

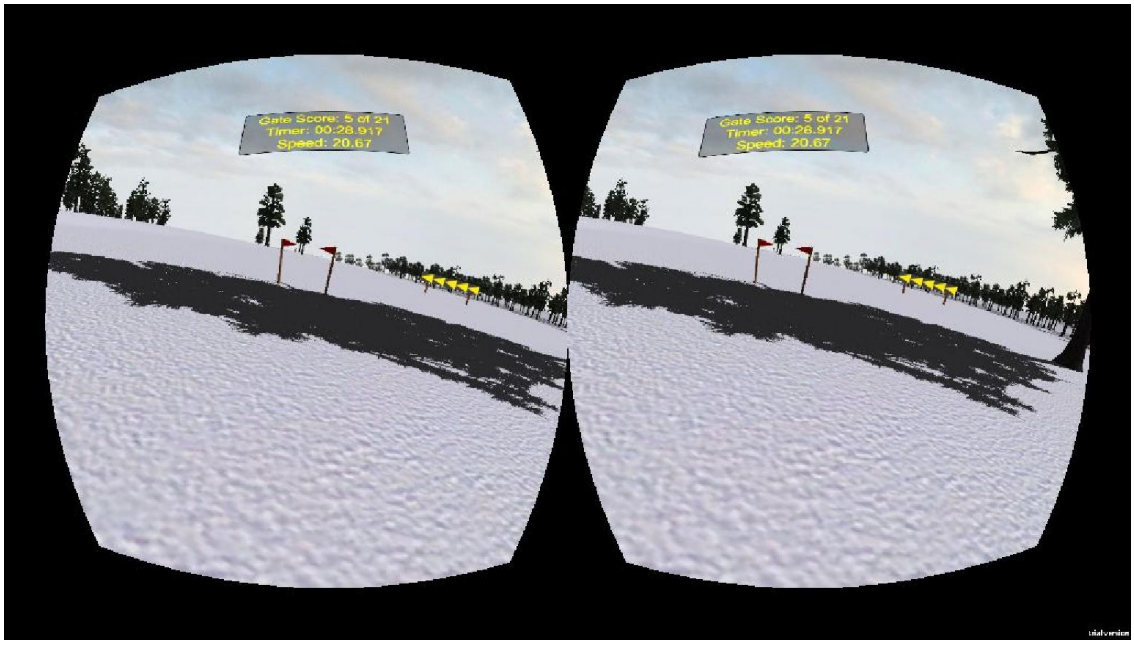


Figure 25 - Real-time detailed tree shadow across the course

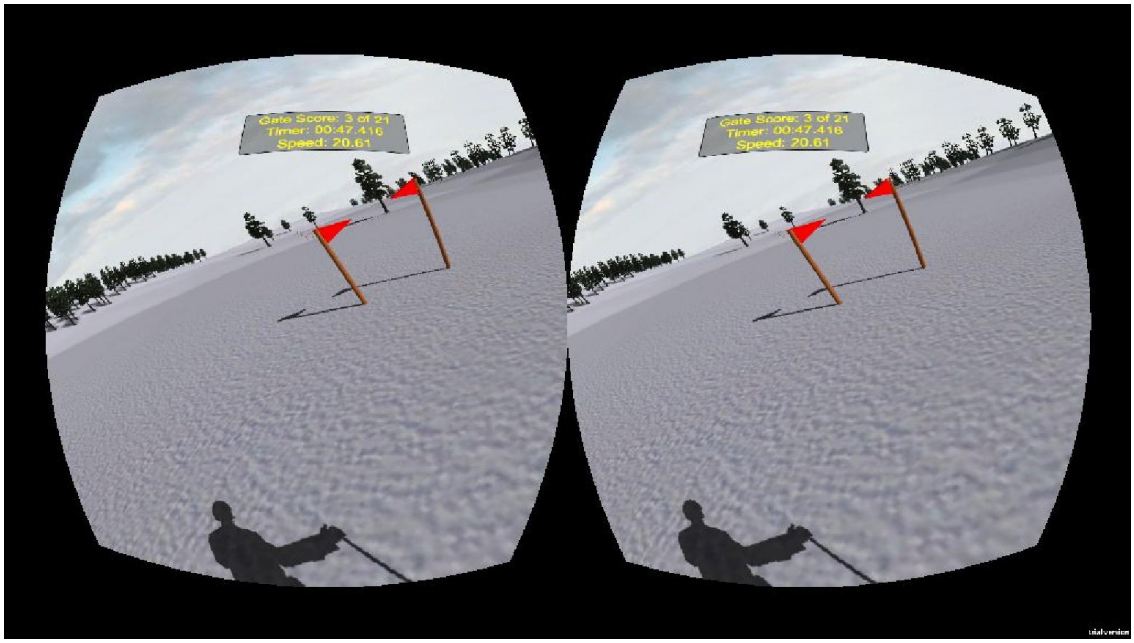


Figure 26 - Making a sharp turn into a gate

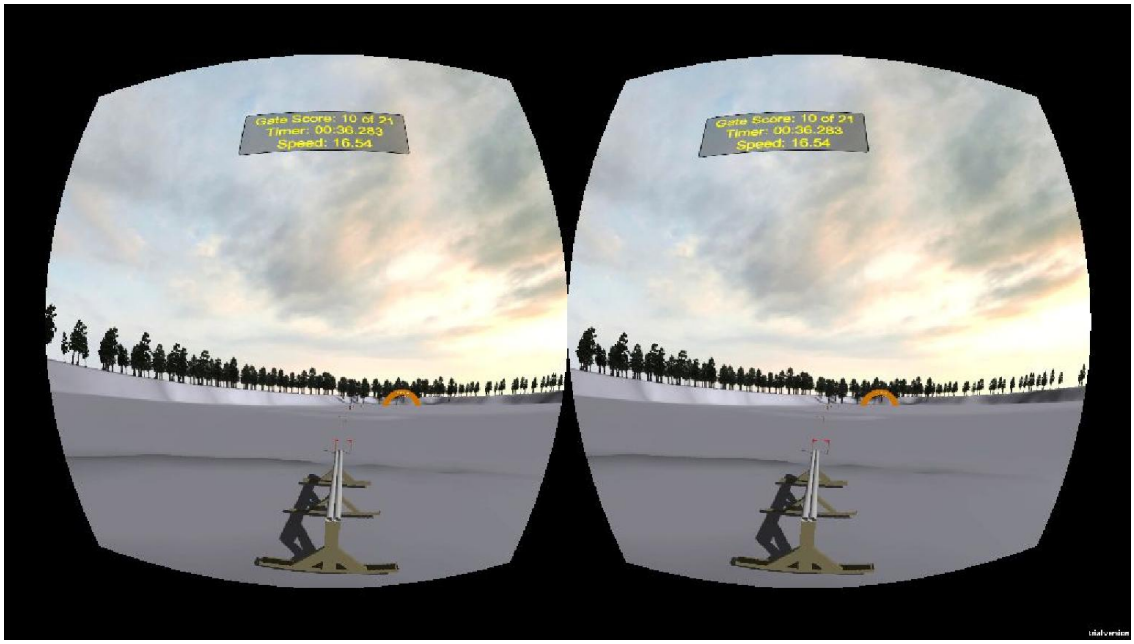


Figure 27 – Jumping onto rails in the Terrain Park

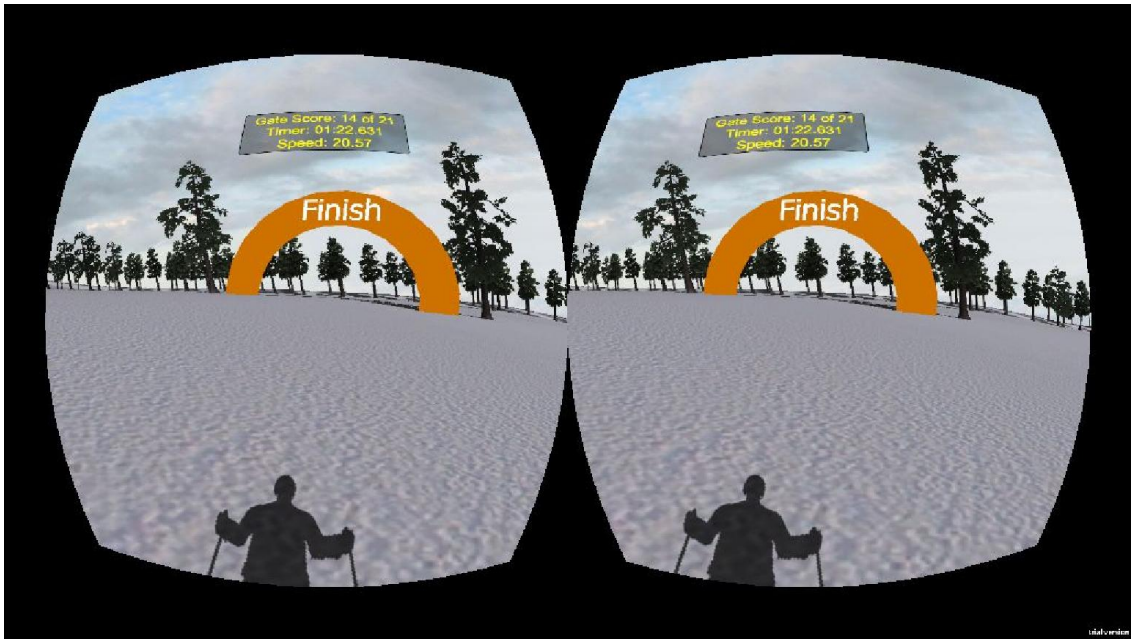


Figure 28 - Approaching the finish line

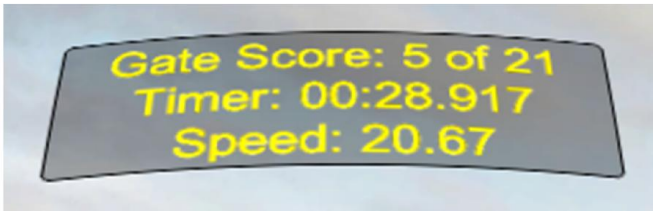


Figure 29 - Close up of the in game UI

The final design of the game UI was changed from the plan in section 3.5.6 to that seen in Figure 29 to respect the users' sense of being in a 3D world. The original concept had UI elements in three distinct positions and this cluttered up the game world, since the UI elements feel as if they are floating in front of the player.

4.2 Images of the game evaluation day

The following photos show the game being played in Snow & Rock, Chertsey. They demonstrate that the primary objectives of the project were met, specifically rendering a first-person view of a slalom skiing experience with movement controlled by head position.

Figure 30 - User testing at Snow & Rock, Chertsey

Figure 31 - More pictures from the user evaluation day

4.3 Videos of the game

Videos of the game in action have been uploaded to a dedicated YouTube channel http://www.youtube.com/user/MarkYoungPortfolio

Description	Link
A video showing the Oculus Rift being held in front of the monitor, making menu choices and skiing a course.	Video 1
A video showing the Oculus Rift being picked up and held in front of monitor, making a number of menu choices and then skiing the advanced Terrain Park course with jumps and rails.	Video 2
A video taken at Snow & Rock of a player performing a single run through of the Val Thorens course.	Video 3
Another player trying the game during the evaluation day at Snow & Rock	Video 4

All players in the videos gave their consent to being recorded.

4.4 Speed and latency

One of the project aims was for the game to run with a minimum consistent frame rate of 60 frames per second. This aim was achieved as shown by the following data taken from Fraps (Fraps 2013) :-

First run
2013-11-24 16:30:55 – Slalom
Frames: 6626 - Time: 111025ms - Avg: 59.680 - Min: 59 - Max: 61

Second run
2013-11-24 16:36:26 – Slalom
Frames: 12561 - Time: 209915ms - Avg: 59.839 - Min: 56 - Max: 62

Figure 32 shows that apart from an occasional single slow frame, caused by triggering a level load, the frame rate held steady at 60fps, which equates to 16.66ms between frames which was the target suggested by the literature review to ensure that head movement to screen movement latency was as small as possible to avoid any perceived lag.

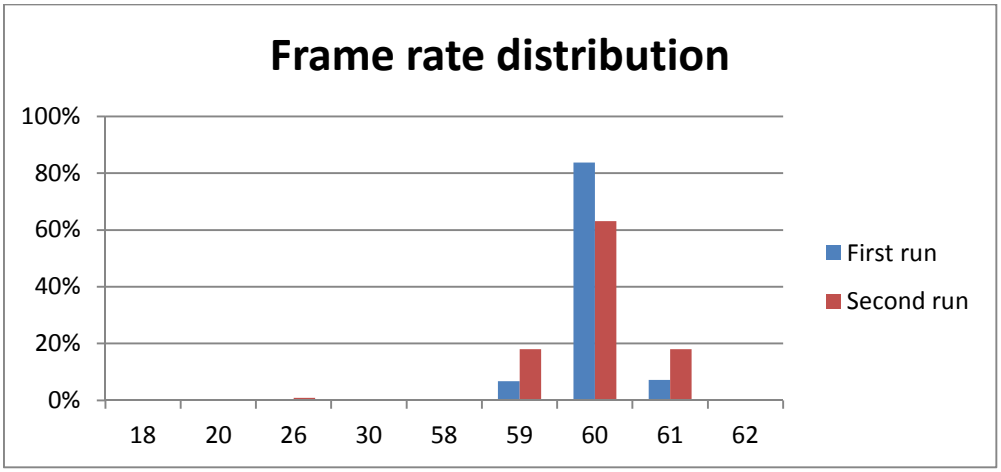


Figure 32 - Frame rate distribution

These figures show that apart from an occasional single slow frame, typically caused by triggering a level load, the frame rate held steady at 60fps, which equates to 16.66ms between frames which was our target to ensure that head movement to screen movement latency was as small as possible to avoid any perceived lag.

4.5 'Look to select' menu system

The original game design was produced before the Oculus Rift was available. A traditional menu system with multiple text screens was envisioned. After the Oculus Rift arrived, and was worn, it had a profound effect on the design because it is so immersive in the way it places the player completely inside the 3D world. It was felt that it was extremely important to respect the player's sense of place in the 3D world even when interacting with the game menu; consequently the menu system was redesigned into a minimal series of floating menu tiles that can be called into existence around the player or dismissed.

Menu choices are made by looking directly at a menu item. To avoid mistakes it is important to have a delay between looking at a menu item and it being chosen. To indicate that a menu item is being looked at a green radial dial appears on top of the menu item and builds from a sliver to a full circle over a two second period, at which point the item is considered to have been selected, this technique is called 'look to select'. This delay would be tedious if a series of choices were required in a row; hence the in-game menu was kept to the bare minimum. More extensive traditional choices such as graphics resolutions and other configuration options would be better handled outside the game using a game configuration screen.

The radial dial system, which indicates that a choice is in the process of being made, was inspired by the Microsoft Kinect (Microsoft 2014) system which requires a player to hold their hand up to move a cursor over a desired menu item and then wait for the menu option to be chosen.

The following pictures show the menu tiles floating in front of the player. Figure 33 shows the scene from a third person view with the player in the process of choosing the 'Val Thorens' menu item. Figure 34 shows the two images rendered for the player's left and right eyes, when wearing the rift each eye sees one image and the composite result is a very immersive experience of standing in front of a series of menu tiles floating around one's head.

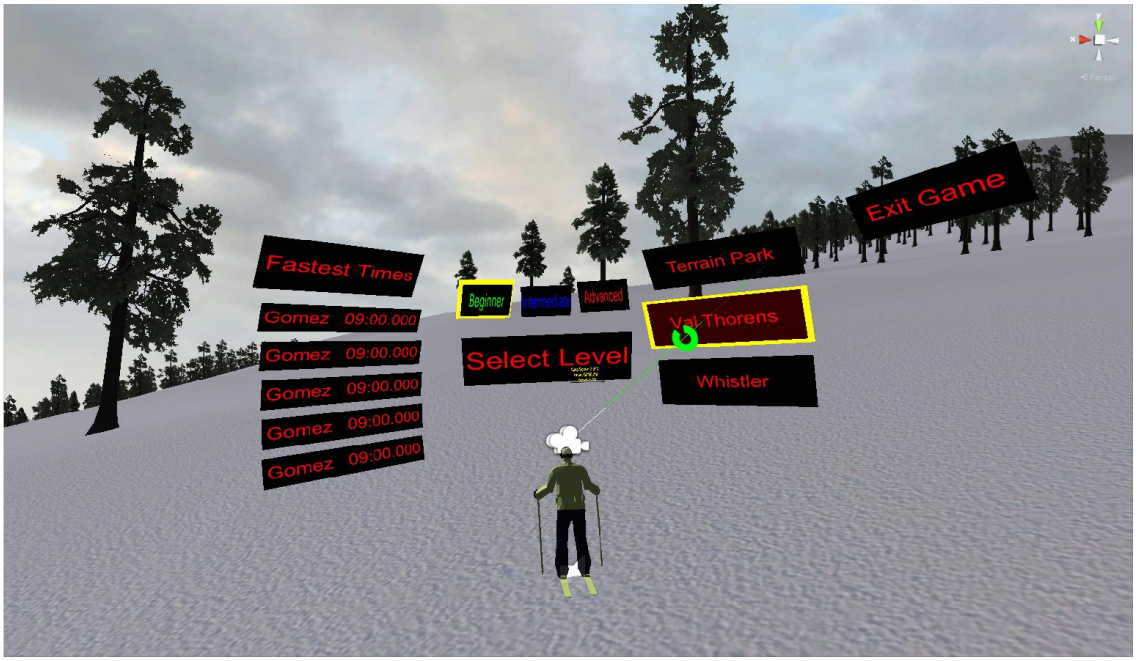


Figure 33 - 'look to select' the Val Thorens menu option

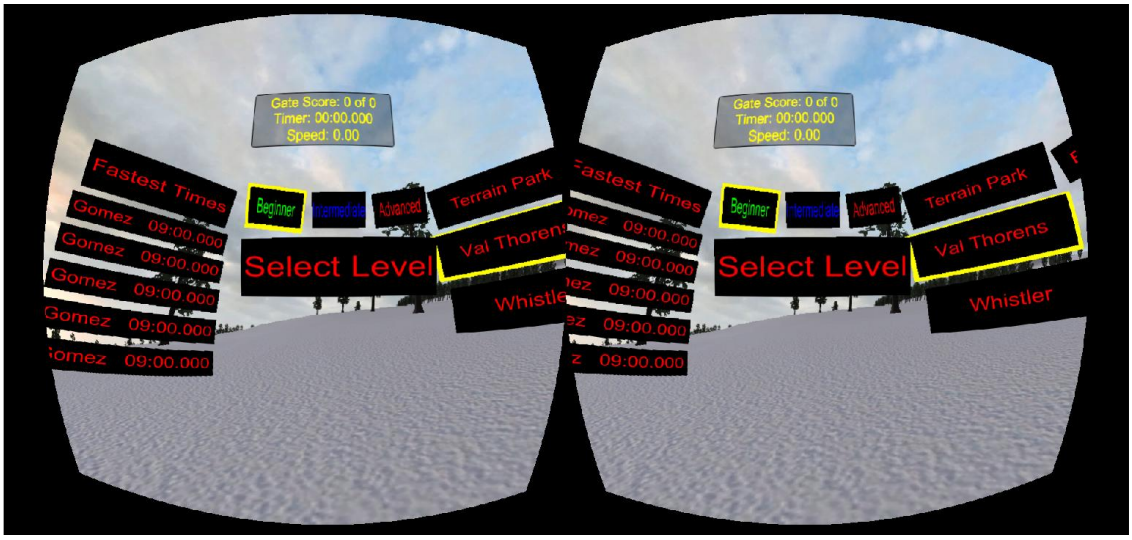


Figure 34 - User view of menu system

97% of people found this menu system intuitive and easy to use.

4.6 UML high level design diagrams

The following Use Case diagram was produced using MagicDraw (No Magic 2013) and shows the simplified 'look to select' menu system built for use within the Oculus Rift.

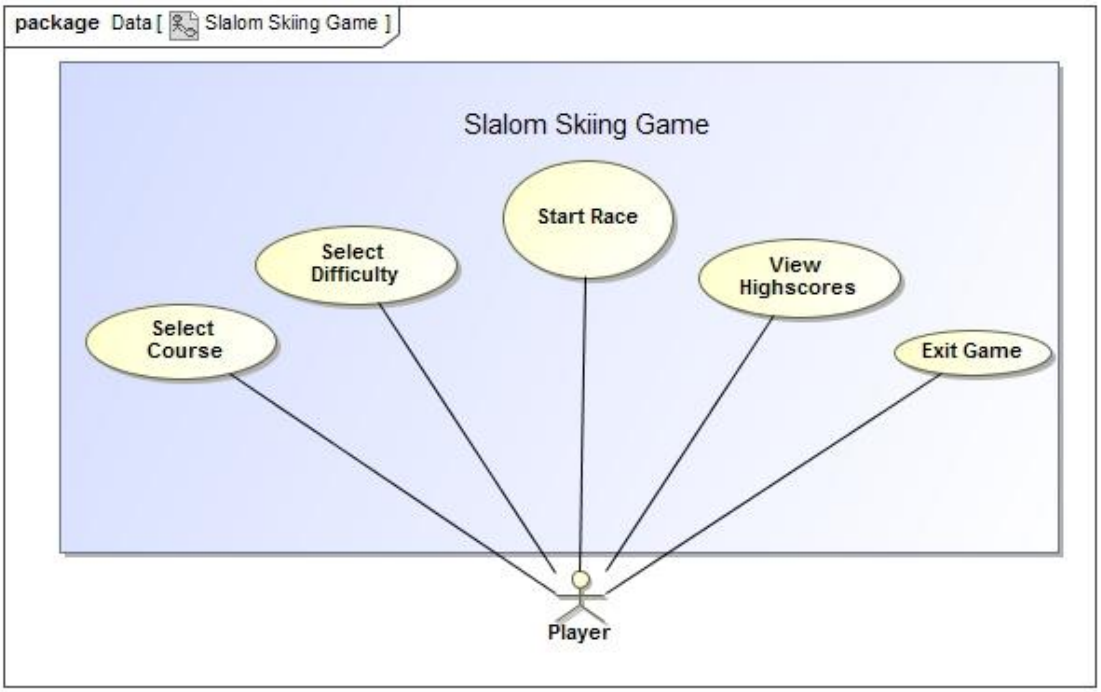


Figure 35 - UML - Use Case diagram

The matching Use Case Specifications can be found in Appendix C.

The following Class Diagram shows the relationship between the gameObjects in Unity. An event based architecture was used giving a loosely coupled and flexible structure to the code.

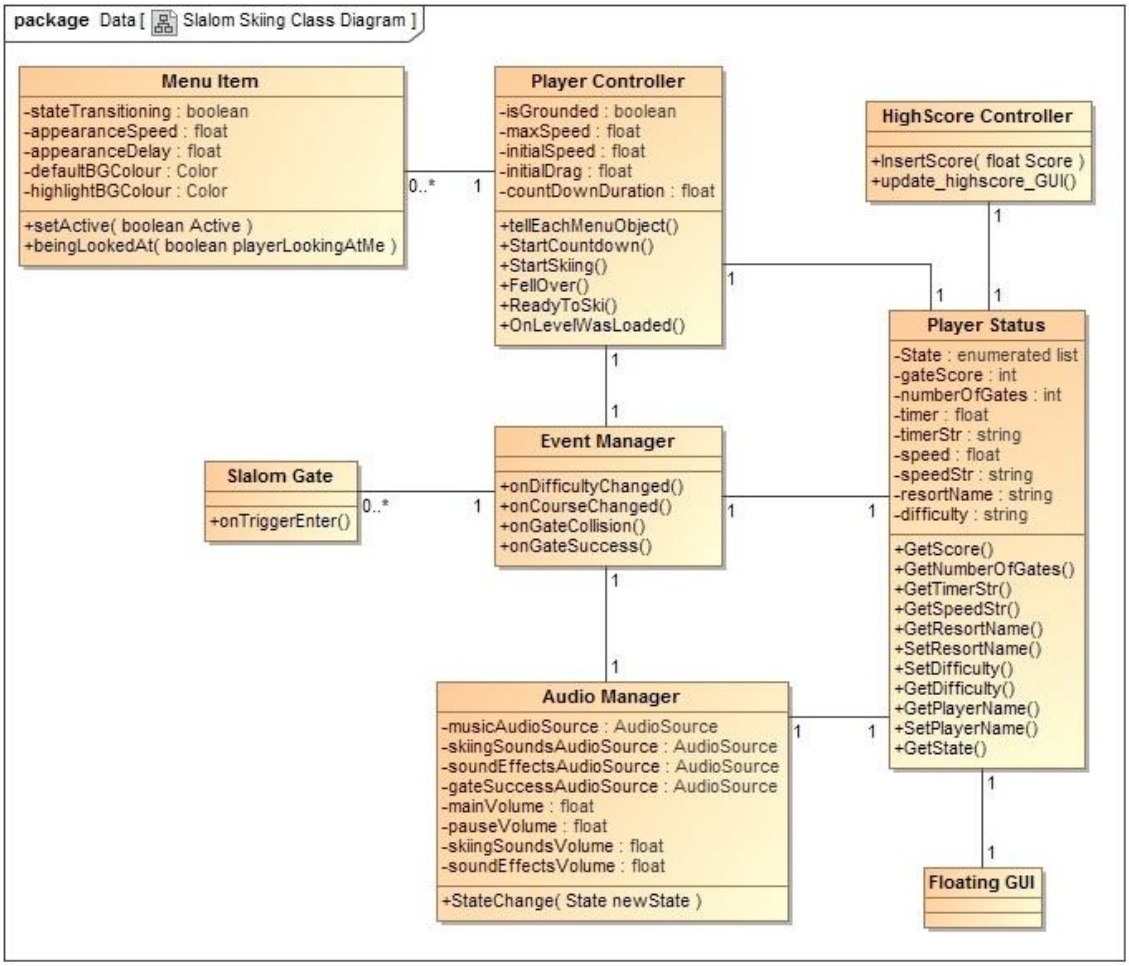


Figure 36 - UML - Class Diagram

4.7 Barrel distortion

Figure 34 above shows the barrel distorted image that is rendered on the display panel inside the Oculus Rift with each eye getting a slightly shifted view of the same scene. The black surrounding areas are invisible to the person wearing the Oculus Rift. As the light from the display panel traverses through the lens inside the Rift, a pincushion distortion occurs that reverses the rendered barrel distortion and causes the scene to be seen properly by the player. The reason the Rift lenses create a pincushion distortion is because they magnify the image to provide an increased FOV (*Oculus SDK Overview v0.2.1* 2013).

The unprocessed image, before the barrel distortion is applied, looks like this

Figure 37 - Unprocessed image prior to barrel distortion

4.8 Entering the player's name into the high score table

If a player gets a high score a way was needed to get their name and place it in the high score table. This presents a problem since when wearing the Rift; the player can't see their keyboard to type their name in. Conceivably the player could be presented with a series of tiles representing each letter of the alphabet along with a delete and enter tile, and then use the 'look to select' technique to spell out their name. This would be tedious.

Given my primary audience for the evaluation of the game were to be customers in Snow & Rock where I would be present whilst they wore the headset and tried the game, I chose the following method to solve this problem. If a high score is achieved a message is displayed on the screen in the black area that is not visible to the player. The message says "Congratulations! High score" and beneath it a text field appears into which a name can be entered. This defaults to the previous player's name, e.g. "Mark", and the high score is immediately placed in the high score table with the name displayed as "Mark?". The name in the field can be edited using the keyboard and when it is correct it can be chosen by pressing Enter. At this point the high score entry is updated. Alternately if the rift player starts another race then the name in the text field is used.

This worked well for the evaluation day. If the game was to be used by a single player alone at home this mechanism would be adapted so that a player profile is chosen by the player when they start the game. This profile would include the Oculus Rift optical settings as well as the player's name, which would then be the default name used in

the game. This would successfully avoid the need to break the immersion of the experience for the player, whilst still putting their name in the high score table.

4.9 Marking the sides of the course

In the original game vision, as inspired by the mood board in section 3.4, the intention was to colour the ground on either side of the suggested path through the slalom gates. During implementation of this visual cue a problem was discovered within the Unity project. The Val Thorens and Whistler levels were built by extracting heightmap data from Google Earth. Once the first scene was built it was duplicated so that alternative courses could be made for the three difficulty levels. It only became apparent later in the development phase that the trees and terrain textures remained constant across the different scenes that had been duplicated. When coloured edges were applied for a course then on switching to a scene containing an alternate gate layout, the course edges were now no longer were in the right positions. So this visual cue was removed from the prototype.

This problem could be overcome by rebuilding the scenes in Unity from the ground up applying the same heightmap to a new terrain; however this would have other difficulties. It would be very hard to get the position of the trees to be the same since the Unity interface does not provide a way to copy the tree positions and textures from one terrain to another. Another problem is that the heightmap data acquired from Google Earth was too irregular in places to give a smooth skiing experience. This was solved by using the terrain smoothing tool. Again to be consistent that smoothing would have to be done on each copy of the terrain. There are ways in which terrain data can be accessed in scripting, it would need to be investigated whether scripting could be used to duplicate the terrain mesh and trees.

4.10 Lighting and shadows

Unity's ability to bake some of the lighting into the scene was used, for instance the shadows cast by the trees on the terrain. This proved to be a much more time consuming and difficult thing to do than expected. It was found that the results produced by the baked lighting were often quite different to those produced by the real-time lighting. In addition sometimes the baking controls did not appear to work as it was understood they should from the documentation.

For instance on one terrain the baked shadows fade out and are replaced by the real-time lighting very smoothly as the player approaches the trees, as shown in Figure 38.

These images demonstrate that the shadow transition occurs over a short distance as the tree is approached, which gives a great effect.

1. Baked shadow

2. Blend of baked and real-time shadow

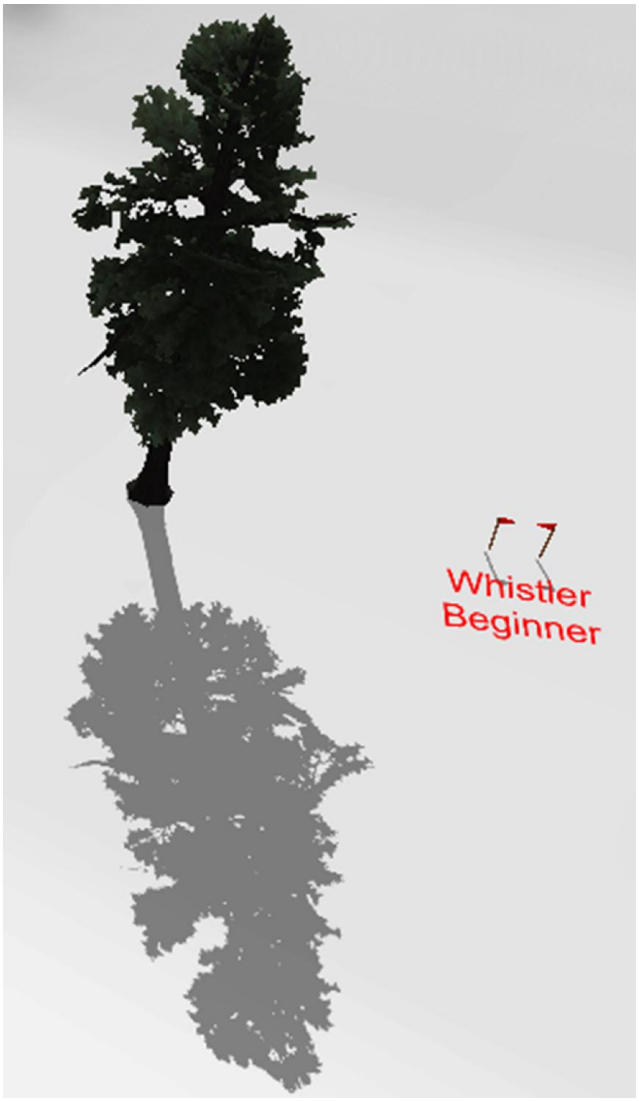
3. Real-time shadow

Figure 38 - Smooth blending of baked and real-time shadows as camera approaches

However another terrain refused to deliver the same result, on that terrain the baked shadow disappears as the player approaches and for a while no shadow is displayed and then as the player gets even nearer the real-time shadow suddenly appears. A lot of time was spent trying to fix this and ultimately the issue was shelved rather than let it impact the rest of the project. These are the sort of time sinks that threaten project management milestones, however resolving such issues leads to a highly polished game. In this case it is suspected that the answers could be found in a book describing how best to utilise Unity's lighting abilities and by a carefully considered request for help on the Unity forum. Problems like this in Unity are sometimes caused by a seemingly innocuous setting.

1. Baked shadow

2. No shadow



3. Real-time shadow

Figure 39 - Poor shadow handling as camera approaches

Notice how much closer the camera has to be for the real-time shadow to be displayed.

4.11 Physics

Unity has a built-in physics engine which was used for the movement and collision detection of the skier on the mountain slopes. A lot of time was spent finding a good collision object shape to attach to the skier to achieve stable and realistic movement. The initial approach, shown in Figure 40, was to use a simple cuboid shape.

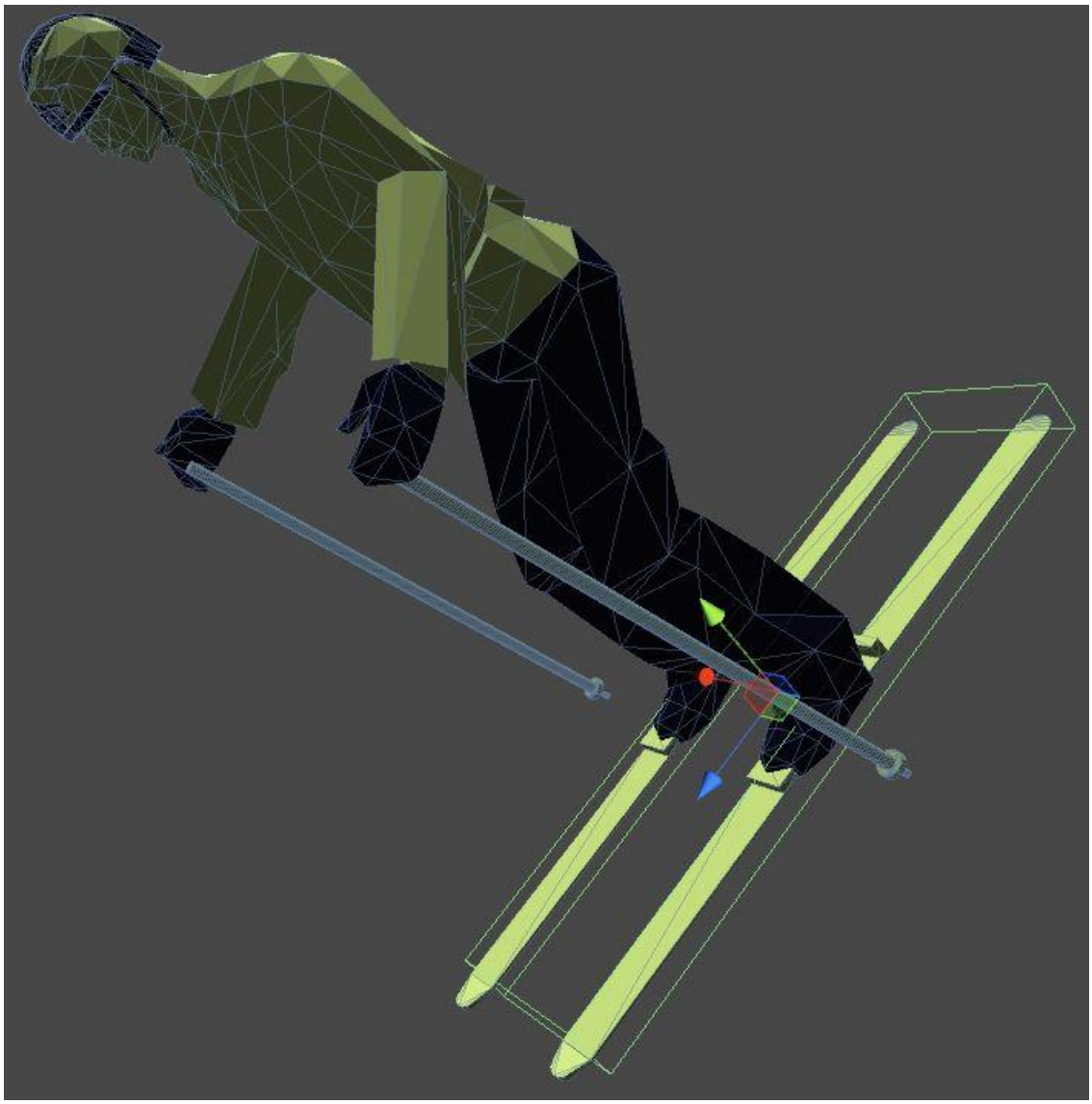


Figure 40 - Skier collider object 1 – rectangle around skis

A simple shape was chosen to minimise collision detection processing. The cuboid is modelled around the skis but is much thicker as it was found that a thin collider can accidentally fall through the terrain in fast moving and steep situations.

A problem with this first approach was stability, it was possible for the player model to fall over and that was not wanted during standard slalom racing, especially with consideration to the player experiencing this motion within the Oculus Rift headset.

The physics engine provides controls to prevent rotation around each axis, however this approach, whilst preventing toppling, is unsatisfactory because the player model no longer accurately rotates to sit on the surface it is travelling over.

For maximum stability the collider shape was expanded further as shown in Figure 41. This collider was very successful and provided realistic physics movement as well as stability.

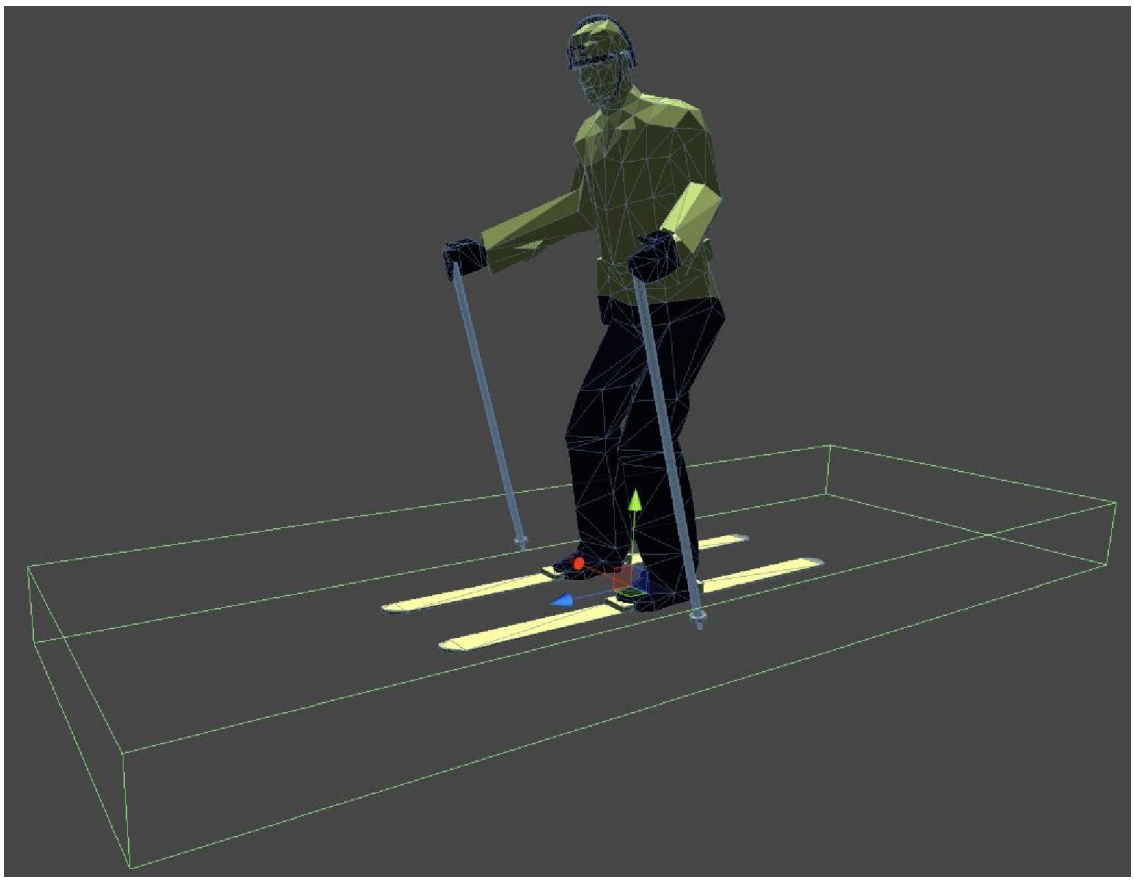


Figure 41 - Skier collider object 2 - huge plank

When the terrain park was added with jumps and dips it became important to allow the player to fall over, but only in a situation where the player did not feel cheated by the experience, since falling over ends the run and requires the player to start again. A problem with the collider in Figure 41 is that if the player does tip over then the side edge is thin and thus the physics engine continues the sideways rotation and the player ends up upside down hanging beneath the landscape from their feet. This was clearly not a desirable situation. So the height of the collider was significantly increased so that the player would fall and lie on their side, which more accurately represents a real life situation, and a large overlapping spherical trigger was added to the head so that when it touches the ground game logic is triggered to perform the end of run mechanics. This final twin collider design is shown in Figure 42.

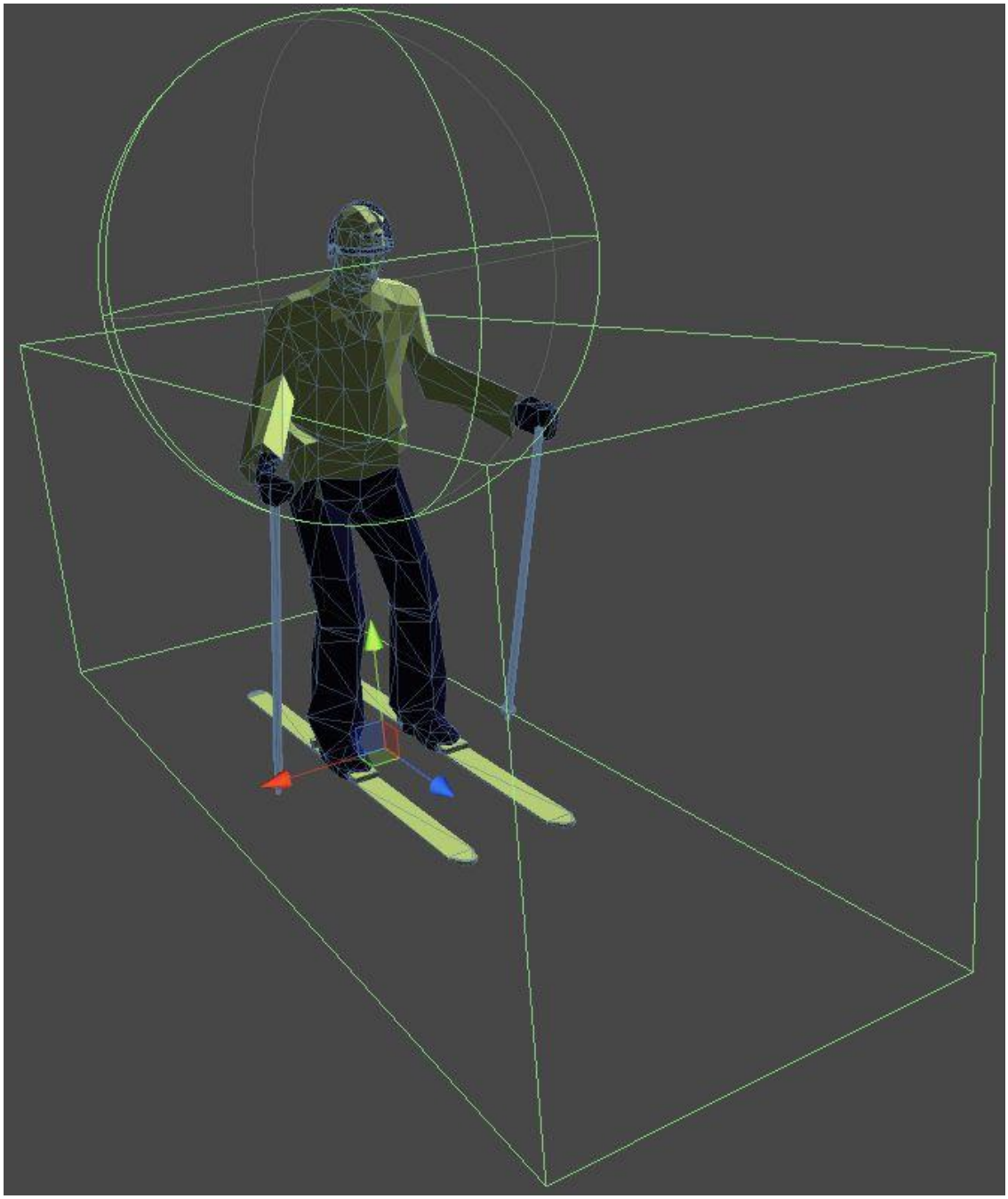


Figure 42 - Skier collider object 3 – twin colliders

Falling over can only occur in extreme situations where a misplaced jump causes the player to tip off a grind rail, as shown in Figure 43.

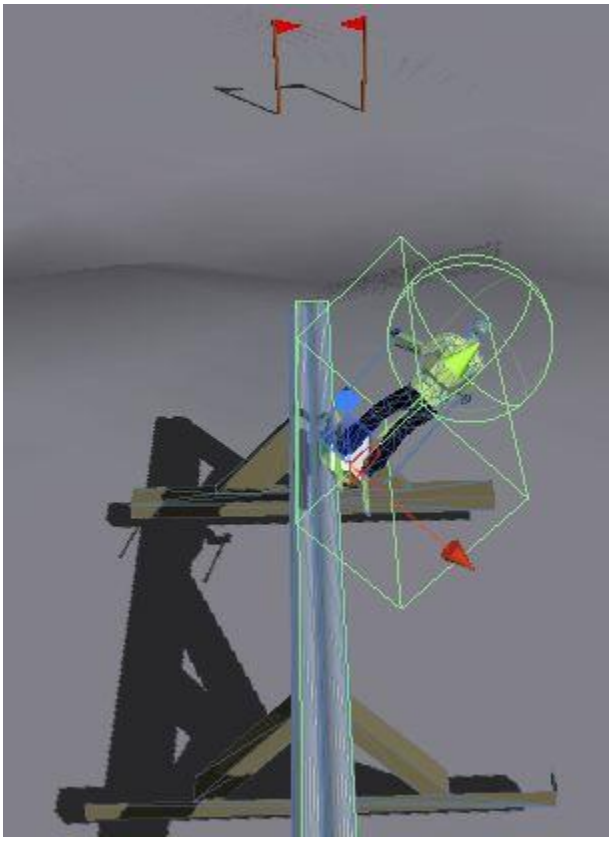


Figure 43 - Skier about to fall over

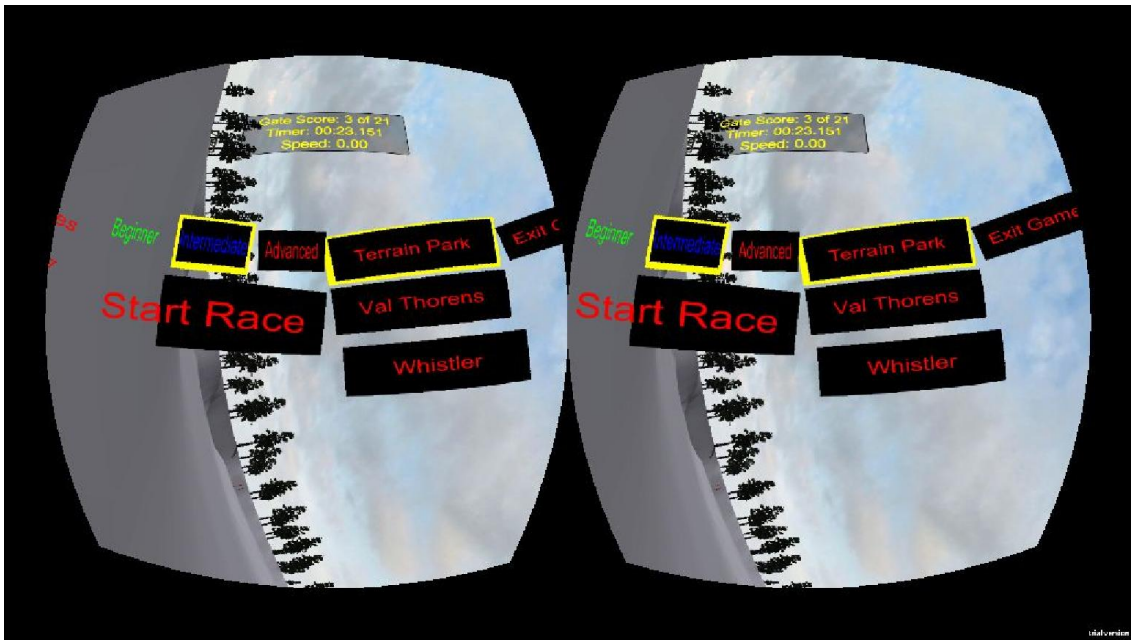


Figure 44 - Player view when fallen over lying on side

4.12 Levels

The game has nine courses in total, three different terrains each with three different course layouts of increasing difficulty. Two of the terrains are based on real mountain slopes and feature traditional slalom skiing courses, the third terrain is built to offer a fun alternative to pure racing and includes jumps, dips and grinding rails around and through which the slalom course is placed.

Figure 45 - Nine courses on three terrains

The Terrain Park is intended for an experienced player. It provides a more varied skiing experience but also opens up the possibility for the player to fall over, which was not possible on the standard slalom courses.

4.13 Summary of key results

In answer to the research questions this project demonstrated that not only is it possible to create a real-time virtual reality slalom skiing game, but that it can be fun too, and the control mechanisms and menu system can be intuitive and easy to use.

This project also demonstrated that the Oculus Rift provides an extremely immersive virtual reality experience at a price point which, whilst not cheap, will be affordable to the game enthusiast. The Oculus Rift developer hardware provides a seamless view into a 3D world and reacts well to physical movement. The evaluation showed that nearly everyone that tried it reported it to be immersive and responsive and for the most part comfortable. The primary weakness of the developer model is the screen panel resolution, 1280x800 giving 640x800 per eye, which is widely expected to be significantly improved in the consumer release. A 1080p version of the headset was demonstrated at E3 in June 2013.

Unity has proven to be an extremely stable and feature rich game development engine. In most cases the free version of Unity will suffice for desktop and web based non-Oculus Rift games, unfortunately at the moment Unity Pro is required to develop for the Rift.

5 Discussion

This chapter reviews the original objectives against the final game produced and details the extent to which they have been achieved. Details are provided where the final project differed from the project plan and some key lessons learnt and recommendations are provided.

5.1 Overall objectives

Taking each research question in turn, as listed in Section 1.2, the key results from this project are reviewed :-

Research Question 1 *Is it possible to design and build a real-time skiing game using the Oculus Rift where the player controls the skier through head motion?*

This project has successfully delivered a real-time virtual reality skiing game controlled entirely by the players head position. The videos in section 4.3 serve to demonstrate the success of this control mechanism. Players that adopted a skiing stance and moved from side to side found that the head position of the Oculus Rift tilted enough to provide the required steering input to play the game.

Research Question 2 *For such a game, what is the user experience in terms of immersion, responsiveness, and fun?*

The evaluation feedback, detailed in Section 3.9.5, showed that 68% of the respondents had 'good fun' and 29% stated they had 'fun'. 97% reported accurate head tracking, 66% moved from side to side whilst playing and 47% reported that their physical steering matched well the movement in the game, with an additional 38% saying that the movement matched some of the time. This is an area where further work should be performed to improve the connection between physical movement and in-game player movement, although it would also bare further study whether players that spent more than three minutes playing the game, and who spent time getting used to the sensitivity of the system, would report better results. Overall 93% found the experience immersive.

Research Question 3 *For such a game, will the player experience disorientation and/or motion sickness?*

50% of the evaluation group reported a sense of disorientation or motion sickness. This is a high percentage. It would bare further study whether this decreased through prolonged exposure to the Oculus Rift as well as identifying the nature of the disorientation or motion sickness.

Research Question 4 *Is it possible to create a menu system controlled from the view inside the Oculus Rift that does not require a mouse or keyboard?*

This project has clearly shown that it is possible to create a menu system which is controlled entirely by head orientation. This system is documented in section 4.5 and videos of it in action can be found in section 4.3.

Research Question 5 *For such a menu system, does the user find the mechanism intuitive?*

The user evaluation found that 97% found the menu control system intuitive and easy to use.

Overall this project has been demonstrated that it is possible to make a fun, responsive and immersive virtual reality skiing game utilising the Oculus Rift alone for full control of the game. The user evaluation results showed that almost every participant had fun and found the game immersive and intuitive.

The main aims of the project as specified in Section 1.3 have all been met, with verifiable results as included in Chapters 3 and 4 of this report.

The rest of this chapter provides some additional detail regarding the challenges faced, some lessons learnt and various choices made whilst working on this project.

5.2 Prototyping using Unity vs. C++

Unity proved to be a very capable game engine. The original proposal included the possibility that the final game would be written in C++ to maintain the stable 60 frames per second that are recommended for a good virtual reality experience. The project was developed on a 6 year old PC with a 4 year old graphics card. Care was taken to implement efficient coding and avoid excessive use of complex 3D models. Ultimately there weren't any speed issues with the game and so there was no need to pursue a C++ version.

The development Oculus Rift has a display resolution of 1280x800. Many developers, myself included, have chosen to render the game at full HD (1920x1080) since that is the expected resolution of the consumer unit. The control box that comes with the Oculus Rift downscales the input signal to match the display resolution. Consequently developing the game to render at 1080p gives the developer the closest setup to the final model. It was noticed that the UI library that comes with the Unity integration package was affected by the PC desktop resolution. This is something that would need to be monitored before final release.

5.3 Speed control via crouching

The initial game design was based on the concept that the player would be asked to physically stand upright at the start of the race and would crouch to begin skiing. In addition it was intended that the depth of the crouch would be monitored and used to further influence the maximum speed. This was to be done using the accelerometer data from the Oculus Rift.

This turned out to be not possible for two reasons: firstly the Unity integration package did not provide access to the accelerometer sensor readings; and secondly when C++ was used to query the accelerometer every 50ms and displayed the sensor values the data values were found to be far too noisy to draw reliable conclusions from. A rolling average was used to smooth the data out, but the results were too erratic to incorporate into the game control system in such a prominent way. A flawed crouch detection system would render the game unplayable.

OculusVR Support were contacted to comment on these findings. They had the following to say with regards to analysing the sensor data

> "Mark,
> Thanks for your interest.
> Inferring positional data from an IMU is very difficult, some would say impossible.

Though in theory you should be able estimate some rough positional movements based on acceleration, there is a lot of noise in the data.
For digital actions like crouching and jumping, you may be able to get something working.
However, I don't believe the Unity integration exposes the raw sensor values.
Sorry I can't be of more help.
Andres”

It is worth noting that the most recent release of the SDK added the ability to access the sensor values within Unity through the GetAcceleration() call. This was tried but the data was still found to be unreliable and so was not included in the final game.

5.4 Evaluation

The evaluation feedback received from members of the public, who were not necessarily gamers, was on the whole very encouraging and rewarding. The majority of participants reported the game to be good fun and immersive with an intuitive menu system.

Half of the participants reported a sense of disorientation or motion sickness and this is clearly an area that bares further study and careful consideration when creating a game to be played in the Oculus Rift.

Care was taken to avoid influencing the feedback by allowing the participant to fill it in anonymously and without oversight. One downside of this approach is that it would have been valuable to discuss some of the comments given for further clarification.

There are two particular areas that would benefit from further discussion with the participant: their feelings with regards to any sense of disorientation whilst wearing the Oculus Rift; and their thoughts on steering movement in the game.

5.5 A question of disorientation

After talking to some friends I have found that some people regarded the experience as disorientating because they found themselves in a realistic representation of a 3D world whilst also knowing they weren’t actually there. I have realised that I didn’t phrase the question sufficiently well to produce the information I was looking for. I would now look to try to determine whether the person felt they could adapt to wearing the Oculus Rift for more than a few minutes, since a truly immersive game would require the player to be able to do that comfortably.

5.6 Steering

The other area where the feedback was 50-50 was the question regarding how well steering movement matched the player's intentions and physical movement. The implemented steering movement is based purely on head tilt, because unfortunately detecting lateral movement was not feasible with the current hardware. Some people adapted to this more easily than others, and that contributed to their feelings on steering. One person in particular commented that tilting the head went against their trained skiing technique in which the upper body should be kept as stationary as possible and, rather like a swan, all the real action and movement should occur in the lower half of the body.

Another area where it was difficult to reach the right steering balance was in dampening existing velocity and applying thrust in the new direction. Many early attempts using standard physics force models led to too much inertial drift and not enough responsiveness to turning forces. It was easy as a player to lose track of how much you were sliding. In an attempt to display the difference between the 'direction of travel' and 'the direction the player was facing', a 2D graphic model of the player as a GUI element was displayed on screen, see Figure 46. It was rotated to show your direction of travel; however it was not very intuitive and didn't prevent the player from getting a sense of disconnection between the direction they were facing and the one they were moving in. Instead heavy dampening was applied through the physics engine and lateral thrusting to increase responsiveness to turning.

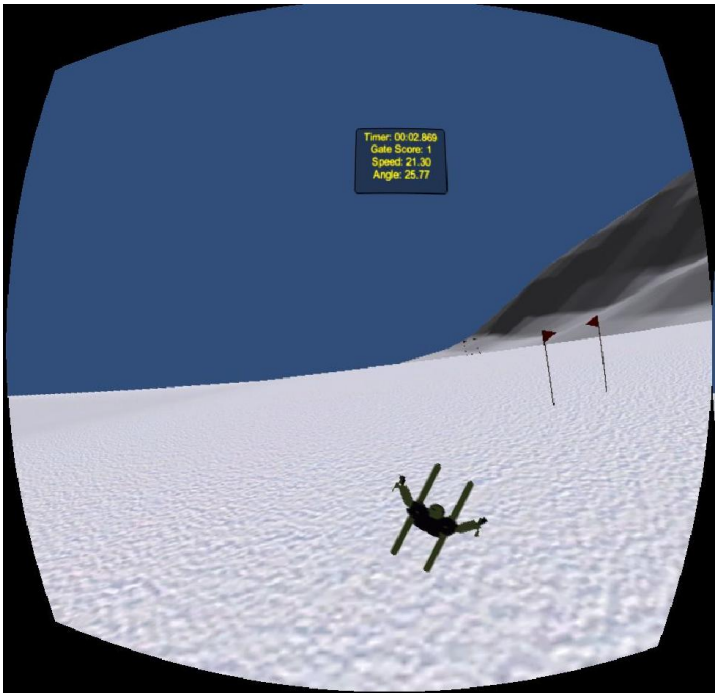


Figure 46 - 2D GUI indicator to show direction of travel

I think the balance that has been struck is not immediately obvious to all players, and perhaps less so to those who have not done slalom skiing. In real slalom skiing you are often at the apex of your turn as you go through the gate - you need to be in order to have enough edging to get to the next gate. Some of this is reflected in my level design, more so at the higher difficulty levels. On the other hand, if you play the game at intermediate or advanced difficulty without forethought for the next gate then you lose too much time and positioning to make it. I think this also contributed to some player's feeling that the steering controls were not responsive enough. It is certainly an area that could benefit from more work, many skiing games just give the player left and right movement, which is less realistic but perhaps easier to adapt to.

5.7 Recommendations

On the basis of delivering this project I would advise fellow developers to prioritise game design around the Oculus Rift. Decide early on whether the player will have any other form of input device such as a game pad. Design the control mechanisms to work well both with the Rift and if appropriate without it. There is always a balance to be struck between the use of non-standard peripherals and default controls, but where possible I would encourage including support of the Oculus Rift and other associated hardware enhancements such as the Omni and Hydra Razer.

Be cautious about the length of time a player will be wearing the Oculus Rift. I suspect that using it will tire the player much sooner than a standard monitor, and some will experience discomfort which will discourage them returning to the game.

The 'look to select' menu system was very successful and well received, as shown in the evaluation feedback. I would recommend using this again. The system could be further enhanced by making the required time spent focused on a menu item be customisable in the player profile, that way users who are fast and confident could reduce the selection time.

I think the level of immersion that is experienced inside the Oculus Rift makes it ripe for games in the horror genre. I have yet to experience this style of game but already a few examples are being well received such as 'Alone in the Rift' and 'Slender: The Arrival'.

Another recommendation would be to avoid a pitfall I fell into. At one stage I decided to see what the project would look like running on an Android phone. I told Unity to build an Android version. This caused Unity to perform a fresh import of all assets and changed the way in which they were optimised. As it turned out the game didn't work at all on the phone. I then switched back to deploying to Windows and found that a

number of assets were either missing or no longer correctly associated with their gameObjects in Unity. The situation was so bad that I had to revert to an early backup of the project directory to recover the situation. This cost me a few days' worth of intricate modifications I'd made to the terrains and course layouts.

5.8 Comparison to Other Work

It is difficult to compare this project to other work since there doesn't appear to be another head mounted virtual reality slalom skiing game to compare it to. The nearest comparison games would be Microsoft Kinect Sports: Season Two, Wii Fit Slalom Skiing and the Alpine Racer 2 arcade machine. Each of those games is a lot of fun and uses hardware-assisted gameplay which requires the player to physically move to drive the game.

In terms of creating an immersive experience I think this project stands up well to these earlier hardware-assisted skiing games, due in large part to the excellent Oculus Rift hardware.

I believe the menu system created in this project is currently unique and very intuitive, although I suspect many similar menu systems will be seen when the Oculus Rift is launched commercially. The main limitation of the menu system is menu item selection speed. I chose a 'look to select' time of two seconds. This gives the user plenty of time to register that a choice is being made and to change their mind, however for repeated menu selections requiring two seconds for each one it could become tedious.

So far I have not seen many Oculus Rift games. Valve has ported Half Life 2 and Team Fortress 2 to use the Oculus Rift. I suspect most Oculus Rift games are in development and awaiting the commercial release of the Rift before making their debut. There are some great demos to experience such as Titans of Space (Drash 2014) and a couple of rollercoasters, and a few simple 5 minute games.

5.9 Summary of lessons learnt

This project has given me the opportunity to spend a lot of time with Unity and learn a lot more about it, which has been a valuable experience. Unity v4 has been completely stable, unlike my experiences of v3, and continues to offer a huge toolkit from which to sculpt a game.

The development process was successful and no major problems were encountered. The agile strategy worked well and gave a continued sense of progression.

As is to be expected in a project of this complexity there were some unforeseen issues, however these were dealt with within the allotted time in the original Gantt chart. Having a Gantt chart in place at the start of the project really helped to keep the work on track. It's very easy to under estimate the time that is required to fully polish a game, areas such as lighting, visual effects, graphic modelling and animation etc. can be exceedingly time consuming. Having a Gantt chart reminding you of your milestone dates and goals really helps to focus productivity and in some cases move on from a tricky problem.

Areas where the work could have gone more smoothly were in the movement mechanics, discussed in section 4.11, and some problems that were unique to Unity such as the lighting and unexpected terrain feature duplication across scenes, discussed in sections 4.10 and 4.9 respectively. Issues such as these are to be expected when using such a comprehensive engine as Unity and are all part of the lessons learned that can be taken forward to the next project.

The literature review was rewarding and interesting, although not always easy to bring the information together to form a coherent narrative.

I also learnt quite a bit about how to run a user evaluation. For this project and audience I strove to keep the length of the questionnaire to a single side of paper. I did not want the person filling it in to be daunted by the size of it. If I was doing it again I would aim to reduce the ambiguity in the questions, and also consider whether to collect the information in the form of a conversational interview with the participant rather than an anonymous multiple choice questionnaire.

6 Evaluation, Reflections and Conclusions

In this chapter the entire project is evaluated, conclusions are drawn and the experience gained summarised. Suggestions for further work are made along with some reflections on the whole experience.

6.1 Project Management

I am pleased with the way this project has turned out. I have been able to keep almost completely on time with the original Gantt chart. I was generous with the allotted time for tasks and that enabled me to conduct some of the research, where appropriate, on different approaches to a problem, or to solve unexpected issues without falling behind the overall schedule. The most critical decision during the project was deciding not to perform any development in C++ with OpenGL, because thankfully Unity was more than capable of delivering the performance needed. It would have been challenging to also build a fully complete game in C++ in the time available.

6.2 Future work

The game needs the attentions of a good artist. It would be nice to include a lot more graphical models to flesh out the world, add some crowds, the sounds of cow bells and many other touches to give it a better sense of realism.

The game design could incorporate more features to encourage players to perfect their technique, such as collecting coins positioned along the route, with bronze, silver & gold reward icons in the leader board to indicate successful collection of 50%, 75% or 100% of the coins. Additional fun could be obtained from speed boosts and two player competitive racing.

A more immersive version of this game could be created by utilising the Wii Balance Board (Nintendo 2013) which could be used to detect the shifting of weight from one foot to the other and use this has the steering input instead of the head tilting. Another immersive add-on would be to use the Microsoft Kinect to analyse body posture and derive steering input into the game.

My ideal goal would be to play the Alpine Racer arcade game using the Oculus Rift as a display and the arcade cabinet foot control mechanism to control the skier, see Figure 1 in section 2.1. I believe that combining Namco's ski like control mechanism with the immersion provided by the Oculus Rift would produce an extremely compelling skiing

game, and perhaps there is a chance that this could happen, since in November 2013 Namco sent out a press release (ArcadeHeroes 2014) revealing that a new Alpine Racer cabinet would be revealed at the 2013 IAAPA trade show (IAAPA 2013).

6.3 Reflection

This project has taught me that tenacity and shear dogged determination is needed at times to drive one through the difficult stages of a project. Unity is a very comprehensive game development environment, although at times it can be difficult to control, as mentioned earlier with lighting. I would like to work with other Unity game creators to share and discuss best approaches to code architecture within the framework it provides. I wouldn't be surprised to find there are better ways to tackle the problems I faced. This is where working in a team opens up the possibilities of learning from the experiences and skills of the other team members.

I would use Unity again. I did not regret it as a choice of game engine. It might be a bit too expensive to use for very small commercial games, but it has a wealth of features and I barely scratched the surface of its capabilities in this project. One can also hope that Oculus Rift integration be introduced into the free Unity offering.

6.4 Conclusion

More than anything this project has shown me that the tools and hardware to make games is widely available to everyone. It doesn't require a large financial outlay to get started, the most critical investment will be your time, and for that you will be rewarded with what you produce and you will find a new found depth of respect for the people who make the games you play.

Go out there and make a game! ☺

7 References

Abrash, M 2012, *Latency – the sine qua non of AR and VR*. Retrieved: July 1, 2013, from http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr

Abrash, M 2013a, *Why VR is Hard (and where it might be going).* Games Developers Conference 2013. Retrieved: June 15, 2013, from http://www.roadtovr.com/2013/03/31/gdc-2013-michael-abrash-virtual-reality-oculus-rift-presentation-slides-4415/all/1

Abrash, M 2013b, *Raster-Scan Displays: More Than Meets The Eye*. Retrieved: July 1, 2013, from http://blogs.valvesoftware.com/abrash/raster-scan-displays-more-than-meets-the-eye

Abrash, M 2013c, *Why virtual isn't real to your brain*. Retrieved: July 1, 2013, from http://blogs.valvesoftware.com/abrash/why-virtual-isnt-real-to-your-brain

Abrash, M 2013d, *Why virtual isn't real to your brain: judder*. Retrieved: July 1, 2013, from http://blogs.valvesoftware.com/abrash/why-virtual-isnt-real-to-your-brain-judder

Alpenglow Games 2013, *Importing terrain from Google Earth into Unity3D in 5 Steps*. Retrieved: August 15, 2013, from http://www.alpenglowgames.com/importing-terrain-from-google-earth-into-unity3d-in-5-steps

ArcadeHeroes 2014, *Namco's New Alpine Racer Arcade Prototype Cabinet Revealed*. Retrieved: January 6, 2014, from http://arcadeheroes.com/2013/11/13/namcos-new-alpine-racer-arcade-prototype-cabinet-revealed

Bilas,S 2002, *A Data-Driven Game Object System.* Game Developers Conference 2002. Retrieved: June 23, 2013, from http://scottbilas.com/files/2002/gdc_san_jose/game_objects_slides.pdf

Blackman, S 2013, *Beginning 3D Game Development with Unity 4: All-in-one, multi-platform game development 2nd Edition*, Apress Media LLC, New York City

Blaha, J 2014, *Diplopia - A Virtual Reality Game to Help Amblyopia and Strabismus*. Retrieved: January 6, 2014, from http://www.diplopiagame.com

Broyad, T 2013a, *Sega R360*. Retrieved: June 14, 2013, from http://www.system16.com/hardware.php?id=844

Broyad, T 2013b, *Ridge Racer Full Scale*. Retrieved: June 14, 2013, from http://www.system16.com/hardware.php?id=537&gid=2693#2693

Broyad, T 2013c, *VR-1*. Retrieved: June 14, 2013, from http://www.system16.com/hardware.php?id=845&gid=2866#2866

Broyad, T 2013d, *Dance Dance Revolution*. Retrieved: June 14, 2013, from http://www.system16.com/hardware.php?id=822&gid=2559#2559

Carmack, J 2013, *Latency Mitigation Strategies*. Retrieved: July 1, 2013, from http://www.altdevblogaday.com/2013/02/22/latency-mitigation-strategies

Clark et al. 2010, *Validity and reliability of the Nintendo Wii Balance Board for assessment of standing balance*. Retrieved: June 23, 2013, from http://0-dx.doi.org.wam.city.ac.uk/10.1016/j.gaitpost.2009.11.012

Creighton, R.H 2011, *Unity 3.x Game Development by Example*, Packt Publishing, Birmingham

Drash 2014, *Titans of Space*. Retrieved: January 2, 2014, from http://titansofspace.net

Fraps 2013. Retrieved: May 27, 2013, from http://www.fraps.com

Freeman et al. 2004, *Head First Design Patterns*, O'Reilly Media Inc, California

Gerasimov & Kraczla 2012, *Unity 3.x Scripting*, Packt Publishing, Birmingham

Google 2013, *Use of images*. Retrieved: December 23, 2013, from https://support.google.com/earth/answer/21422?hl=en&ref_topic=2376153

IAAPA 2013, International Association of Amusement Parks and Attractions Expo 2013. Retrieved: January 6, 2014, from http://www.iaapa.org/expos/iaapa-attractions-expo-2013/Home

IAM 2013a, *International Arcade Museum - Hard Driving'*. Retrieved: June 14, 2013, from http://www.klov.net/game_detail.php?game_id=8072

IAM 2013b, *International Arcade Museum - Final Furlong*. Retrieved: June 14, 2013, from http://www.klov.net/game_detail.php?game_id=7796

IAM 2013c, *International Arcade Museum – Police 24/7*. Retrieved: June 14, 2013, from http://www.arcade-museum.com/game_detail.php?game_id=9065

IAM 2013d, *International Arcade Museum – Alpine Racer 2*. Retrieved: June 14, 2013, from http://www.klov.net/game_detail.php?game_id=6870

INM375 2012, 'Spiral Development Model', INM375 Game Development Process module 2012, City University London, lecture 2, slide 38.

Kyaw & Swe 2013, *Unity 4.x Game AI Programming*. Packt Publishing, Birmingham

Lee et al. 2012, *Development of an augmented reality-orientated game system for stroke rehabilitation assessment*. Biomedical Engineering-Applications Communications. Volume: 24 Issue: 5 Pages: 435-445. DOI: 10.1142/S1016237212500391 Published: OCT 2012.

Merhi et al. 2007, *Motion sickness, console video games, and head-mounted displays.* Retrieved: June 23, 2013, from http://www.ncbi.nlm.nih.gov/pubmed/17915607?dopt=Abstract

Menard, M 2012, *Game Development with Unity*, Course Technology Cengage Learning, Boston

Microsoft 2014, *Microsoft Kinect*. Retrieved: January 1, 2014, from http://www.microsoft.com/en-us/kinectforwindows

Nintendo 2013, *Wii Fit*. Retrieved: June 26, 2013, from http://www.nintendo.com/games/detail/hoiNtus4JvIcPtP8LQPyud4Kyy393oep

No Magic 2013, *MagicDraw*. Retrieved: November 18, 2013, from http://www.nomagic.com/products/magicdraw.html

OculusVR 2013. Retrieved: April 25, 2013, from http://www.oculusvr.com

Oculus SDK Overview v0.2.1 2013. Retrieved: April 25, 2013, from https://developer.oculusvr.com/documents/Oculus_SDK_Overview_0.2.1.pdf

Peli E, 1998, The visual effects of head-mounted display (HMD) are not distinguishable from those of desk-top computer display. Retrieved: June 24, 2013, from http://0-dx.doi.org.wam.city.ac.uk/10.1016/S0042-6989(97)00397-0

Project Holodeck 2013. Retrieved: June 14, 2013, from http://www.projectholodeck.com/system

RoG et al. 2013, *50 Greatest Arcade Cabinets in Video Game History!* Retrieved: June 14, 2013, from http://www.i-mockery.com/minimocks/50arcadecabinets

Rollings & Morris 2004, *Game Architecture and Design: A New Edition.* New Riders Publishing, Indianapolis.

Scooter Software 2013. *Beyond Compare by Scooter Software*. Retrieved: November 29, 2013, from http://www.scootersoftware.com

SkyTechSport 2013, *Breakthrough Winter Sports Simulation Technology.* Retrieved: May 28, 2013, from http://www.skytechsport.com/ski-simulators/breakthrough-technology

Smith & Queiroz 2013, *Unity 4.x Cookbook*, Packt Publishing, Birmingham

Solina F, Batagelj B, Glamočanin S 2008, *Virtual Skiing as an Art Installation,* Retrieved: May 5, 2013, from http://eprints.fri.uni-lj.si/241/1/ELMAR2008-virtual_skiing_-_Final.pdf

SugarSync Inc 2013. *SugarSync – File Sync & Online Backup*. Retrieved: November 29, 2013, from https://www.sugarsync.com

Unity Technologies 2013, *What is Unity and what can I do with it?* Retrieved: June 23, 2013, from http://unity3d.com/pages/create-games

Untouchable Events 2013, Alpine Racer virtual reality simulator at Untouchable Events. Retrieved: June 21, 2013, from http://www.untouchableevents.com/party-activities-party-favors.html#tabs-3

Valve 2013. *Steam*. Retrieved: June 23, 2013, from http://www.steampowered.com

Virtuix 2013. *Omni.* Retrieved: June 22, 2013, from http://www.virtuix.com

Wittaybundit, J 2011, *Unity 3 Game Development Hotshot,* Packt Publishing, Birmingham

West, M 2007, *Evolve Your Hierarchy.* Retrieved: June 23, 2013, from http://cowboyprogramming.com/2007/01/05/evolve-your-heirachy

Whiting, N 2013, *Integrating the Oculus Rift into Unreal Engine 4.* Retrieved: June 30, 2013, from http://gamasutra.com/blogs/NickWhiting/20130611/194007/Integrating_the_Oculus_Rift_into_Unreal_Engine_4.php

Appendix A: Project Proposal for MSc in Computer Games Technology

Name:	**Mark Young**
E-Mail Address:	**mark_young@hotmail.com**
Contact Phone number:	**01932829593**

Project Title:	**A skiing game utilising Oculus Rift head tracking to intuit body position and infer slalom skiing movement**
Supervisor:	**Dr. Greg Slabaugh**
Date:	**16th May 2013**

Introduction

This is a design and build project to create a slalom skiing game using the Oculus Rift. The Oculus Rift is an exciting new way to experience games. It is a virtual reality head mounted display with movement sensors that is due to be available for consumer release in 2014. This means that the player can be immersed into the game world such that the view presented to them aligns with their head orientation.

The principle aim of this project will be to put the player into a first person perspective skiing down a mountain utilising the Oculus Rift's head tracking ability to allow the player to look around. A secondary goal will be to attempt to intuit the player's physical position and apply that to the in-game character movement.

Project description

The Oculus Rift is a brand new technology which is just now being released to developers before the expected consumer release in 2014, consequently it is barely past the development stage. Whilst the developer model hardware is now fixed, and being shipped, the consumer model is yet to be finalised, at the very least it is expected to have higher resolution screens.

There is much to learn about the requirements of human perception whilst wearing a head mounted display. The Oculus Rift SDK overview document [1] takes pains to state that for the best virtual reality experience, the frame rate must hold consistently at 60fps with 'motion-to-photon' latency of no more than 60ms and ideally less. These are critical factors in achieving an immersive experience.

There will be a large number of technical challenges to overcome in this project. As the Oculus Rift is only now starting to get into the hands of developers it is a very fluid time to evaluate which language and approach to use to code this game. An early evaluation of the documentation and SDK will be essential to decide between the use of C++ using DirectX or OpenGL, and Unity. Currently the software development kit (SDK) contains sample code written for DirectX only, whilst this author's experience is with OpenGL. The Oculus Unity integration package has just been released, so at this stage it appears most prudent to plan to use Unity to produce a prototype of the game and look to adopt OpenGL later if feasible.

Aims and objectives

	Objective	Priority	Testable Result
1	Literature review of hardware enhanced skiing games and head movement player controls	High	Inclusion into the Project Report
2	Scene rendering to provide a first-person view that changes based on head movement	High	Images in Project Report and video of game in action
3	Ability to ski down a slope using head tracking input to control speed and make slalom turns	Medium	Video of game being played
4	Achieve a consistent frame rate of 60 frames per second (FPS)	Medium	Benchmarked using FRAPS [4]
5	Player evaluation including, but not limited to: immersion; responsiveness; movement control feedback; motion sickness and fun.	High	Feedback from players recorded and summarised in Project Report

Project Beneficiaries

The primary beneficiary will be the author of this project who will gain wide ranging experience in coping with new technology and incorporating that with both Unity and potentially OpenGL with C++. Additional beneficiaries will be players of this game, which may include complete ski beginners getting a feel for skiing. Finally lessons learned from this project may be informative to other developers considering similar work using the Oculus Rift. One way in which the experiences from this project will be shared with others is via the Oculus Developer forum website.

Methodology

This is a design and build project with evaluation both of the technical aspects of the use of a head mounted display as well as the user perceptions of it. Using the Spiral model software design process [5][6] the project will be built in a series of prototype stages in an iterative fashion, where each iteration will be a milestone on the road to the final game experience.

The project will include the following:

1. Prototyping using the Unity engine for the best chance at producing a viable sample game
2. If time permits a C++ and OpenGL version of the game will be built. The main reason to adopt this path will be for performance and ultimate developer control.
3. A game design will be produced to document the expected player experience
4. The game will be built using a Model View Controller (MVC) loosely coupled flexible game architecture.
5. An iterative approach will be taken to the development of the skiing game and the head mounted control system. The software will be fleshed out in stages including:
    a. Static scene generation
    b. Movement down a slope
    c. Sensor input from Oculus Rift to map view to head orientation
    d. Oculus Rift sensor input to define speed of movement and sharpness of turns
    e. The addition of gameplay mechanisms to encourage repeated play of the game such as quickest time, highest score and fastest speed achieved, with associated leader boards to encourage players to compete.
6. An evaluation of player experiences will be conducted, ideally via the cooperation of a local skiing shop or dry ski slope. My intention is to approach the shop and see if they will let me set up the Oculus Rift and slalom skiing game on their premises for a day. Adult members of the public will be encouraged to fill in a questionnaire and provide feedback on their experience.

Theory

The Oculus Rift creates some additional and unique challenges on top of normal game design and implementation. The game engine must render two game world views, side by side, one from the position of each eye. The distance between the eye pupils of an average human, known as the interpupillary distance (IPD), is 64mm but can range between 54mm to 72mm [1]. For the best player experience the game rendering should be adjusted to match the

players IPD. Setting this value is something that needs to be tied to a player profile to facilitate quick changes between players.

The Oculus Rift does not come with a calibration utility, nor include an example of such a system in their SDK. Typically players will not know their own IPD. Measuring it with a ruler is very difficult and a visit to an optometrist is inconvenient. Valve have solved this problem by including an IPD calibration system into their popular team based multiplayer first person shooter Team Fortress 2 [3] which now incorporates support for the Oculus Rift.

The Oculus Rift contains lenses that magnify the image to increase the field of view (FOV). This creates a pincushion distortion of the image, consequently the game engine must counter this by applying a barrel distortion to each rendered image. It is suggested that this be done by a pixel shader.

In order for the player to relax, suspend disbelief and enjoy the virtual reality (VR) simulation the game engine must render at least 60 frames per second (FPS) with vsync enabled and without dropping frames, and the latency from head movement to updated image position being seen must be as low as possible. The Oculus SDK Overview states “Although 60ms is a widely cited threshold for acceptable VR, at Oculus we believe the threshold for compelling VR to be below 40ms of latency. Above this value you tend to feel significantly less immersed in the environment.” [1]. A considerable portion of this time is in the hardware, hence the aim for the game engine to use only 16.6ms (60 FPS).

Research Context

Previous research has been performed to recognise and react to physical body position for the purposes of virtual skiing. In 2008 an art installation was built in Ljubljana, Slovenia to allow a user to ”immerse himself into the skiing sensation without using any obvious hardware interfaces” (*Oculus SDK Overview v0.2.1* 2013). This was done by creating a physical ski slope scene, complete with artificial snow, skis and poles for the user to use, in a room facing a wall onto which a virtual ski slope scene is projected. A small camera in front of the user monitored body posture and used this to drive in game movement: left and right turns and crouching to increase speed. This approach to virtual skiing focused highly on computer vision to determine skier movement whereas this project aims to envelope the player in the game world and utilise head position to control player movement.

Companies like SkyTechSport.com [7] are combining high tech sports training equipment with huge virtual reality projected displays to enable athletes to train on GPS mapped replicas of [Giant] Slalom race courses in preparation for major events.

I have not been able to find a previous attempt to make a head mounted display virtual reality skiing game.

Evaluation

Feedback will be sought from a small sample of players to solicit their opinion on the following areas

1. Immersion. How immersive was the experience, did the virtual reality display render a convincing world?

2. Visual responsiveness. Was the rendering able to keep pace with player head movement sufficiently to enable the player to achieve suspension of disbelief?
3. Control responsiveness. Was the player able to accurately control the in-game skiing movement through their own physical position?
4. Motion Sickness. Did the player experience any discomfort or disorientation?
5. Fun. Was the experience enjoyable and fun?

Deliverables

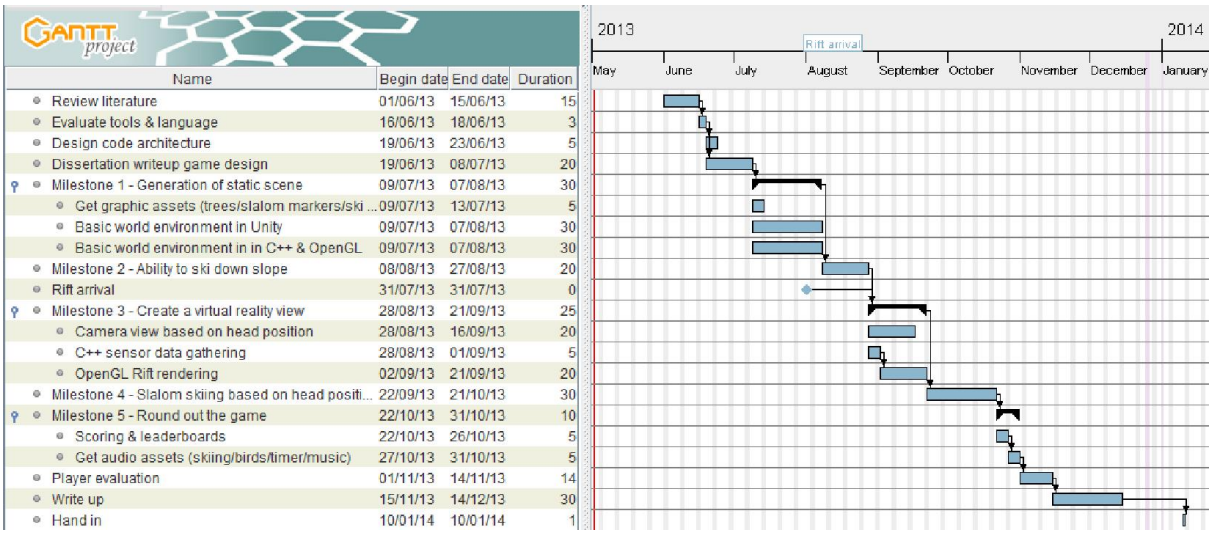
Deliverables will include

1. Game design documentation
2. Game architecture documentation
3. The game itself, delivered as an executable
4. A video of the game being played
5. An evaluation of player experiences and feedback using the Oculus Rift both as a virtual reality display and as an implicit controller

Feasibility

It is very easy to plan to incorporate more and more features and game design ideas, however with the limited time available and the cutting edge technology being investigated my goals are quite minimal on the game front. As listed in the aims and objectives section I am primarily aiming to get a single slalom slope working and evaluated.
I believe that the Unity game engine makes this project feasible because it can be an extremely powerful and flexible game prototyping tool, and despite this author's minimal experience with Unity I believe there is enough time to get the basics in place.
Ideally I would like to get the game also running in C++ using OpenGL.

Project Time Plan



Name	Begin date	End date	Duration
Review literature	01/06/13	15/06/13	15
Evaluate tools & language	16/06/13	18/06/13	3
Design code architecture	19/06/13	23/06/13	5
Dissertation writeup game design	19/06/13	08/07/13	20
Milestone 1 - Generation of static scene	09/07/13	07/08/13	30
Get graphic assets (trees/slalom markers/ski ...	09/07/13	13/07/13	5
Basic world environment in Unity	09/07/13	07/08/13	30
Basic world environment in in C++ & OpenGL	09/07/13	07/08/13	30
Milestone 2 - Ability to ski down slope	08/08/13	27/08/13	20
Rift arrival	31/07/13	31/07/13	0
Milestone 3 - Create a virtual reality view	28/08/13	21/09/13	25
Camera view based on head position	28/08/13	16/09/13	20
C++ sensor data gathering	28/08/13	01/09/13	5
OpenGL Rift rendering	02/09/13	21/09/13	20
Milestone 4 - Slalom skiing based on head positi...	22/09/13	21/10/13	30
Milestone 5 - Round out the game	22/10/13	31/10/13	10
Scoring & leaderboards	22/10/13	26/10/13	5
Get audio assets (skiing/birds/timer/music)	27/10/13	31/10/13	5
Player evaluation	01/11/13	14/11/13	14
Write up	15/11/13	14/12/13	30
Hand in	10/01/14	10/01/14	1

Risk Assessment

Provide a Risk table that identifies all potential objectives with values ranging from 1 to 10.

	Task	Severity of failure	Risk of failure	Severity X Risk	Mitigation
1	Literature Review	1	1	1	Do literature review first
2	Evaluation of development tools	5	1	5	Seek knowledge of others using an Oculus Rift via forums
3	Oculus Rift availability	10	1	10	Borrow one from full time student
4	Produce Unity prototype	10	1	10	SDK released and Unity Pro plugin released
5	Produce OpenGL version	3	4	12	Unity version may well be good enough for this project
6	Incorporation of head tracking data into game	4	2	8	Game could be played via controller instead
7	Oculus Rift integration issues causing delays	6	2	12	Use of the Unity Rift camera controller, reduced aims on size of game.

References

[1] Oculus SDK Overview v0.2.1, https://developer.oculusvr.com/documents/Oculus_SDK_Overview_0.2.1.pdf, downloaded 25th April 2013.

[2] Virtual Skiing as an Art Installation, http://eprints.fri.uni-lj.si/241/1/ELMAR2008-virtual_skiing_-_Final.pdf, downloaded 5th May 2013.

[3] Calibrating your Inter-Pupillary Distance, http://wiki.teamfortress.com/wiki/Oculus_Rift_User_Guide#Calibrating_your_Inter-Pupillary_Distance, downloaded 16th May 2013.

[4] FRAPS real-time video capture and benchmarking software, http://www.fraps.com/, accessed 27th May 2013.

[5] Spiral model, http://en.wikipedia.org/wiki/Spiral_model, viewed 28th May 2013

[6] Spiral Development Model, INM375 Game Development Process module 2012, City University, lecture 2, slide 38.

[7] SkyTechSport, http://www.skytechsport.com/ski-simulators/breakthrough-technology, viewed 28th May 2013

Appendix B: Example code

This is the GameManager.cs C# script. This script uses the Singleton Design Pattern to ensure it is the only instance of itself running. It is then responsible for creating a Player gameObject if none exists. The reason for this is explained in Section 3.6.1

```
using UnityEngine;
using System.Collections;

// The purpose of this script is to check if there is a Player object
// after level load and if not instantiate one and position the player
// correctly at start and after each level load.

// NOTE: If we have a permanent player object we can not load a level
//       that has one as an active gameObject on load, even if it
//       immediately destroys itself, because this causes the Rift
//       camera to freeze and the game to hang.
//       Cardinal rule) Never load a level that contains a player game
//                      object if one is already instantiated and set
//                      to DoNotDestroyOnLoad

// The GameManager applies the Singleton Design Pattern

public class GameManager : MonoBehaviour {

    public static GameManager _instance = null;
    public static GameManager Instance { get { return _instance; }}

    public GameObject playerPrefab;
    private GameObject player = null;

    void Awake() {
        // Enforce us as a Singleton gameObject (if we were beaten to it, then destroy ourself)
        if (_instance != null) {
            Destroy(gameObject);
        } else {
            _instance = this;
        }
        DontDestroyOnLoad(gameObject);

        PlayerController.playerStateChangeEvent += StateChange;

        CreatePlayerIfMissing();
    }

    void OnLevelWasLoaded() {
        CreatePlayerIfMissing();
    }

    // Check for a Player gameObject, create one if none found
    void CreatePlayerIfMissing() {
        player = GameObject.FindWithTag("Player");
        if (player == null) {
            player = Object.Instantiate(playerPrefab) as GameObject;
            DontDestroyOnLoad(player);
        }
    }

    // Find the gameObject tagged as 'Respawn' and position the player there
    public void PositionPlayer() {
        GameObject playerStart = GameObject.FindWithTag("Respawn");
        if (playerStart != null) {
            player.transform.position = playerStart.transform.position;
            player.transform.localEulerAngles = playerStart.transform.localEulerAngles;
        } else {
            Debug.Log("No gameObject found tagged as 'Respawn'");
        }
    }

    // Ensure player correctly positioned when state set to 'ready to ski'
    void StateChange(PlayerStatus.State newState) {
        if (newState == PlayerStatus.State.ReadyToSki) {
            PositionPlayer();
        }
    }
}
```

Figure 47 - Source code - GameManager.cs

This is the MenuItemController.cs C# script. This script is attached to a menuItem gameObject. It uses a C# delegate to subscribe to a menuItemsActivationEvent event which triggers the menuItem flying into view or flying away. (pubsub Design Pattern)

```
using UnityEngine;
using System.Collections;

// Attach this script to an object (e.g. a MenuItem prefab) which will
// act as a float menu item to hover around the players head when the
// menu system is triggered.
//
// The renderer is enabled/disabled to efficiently control visibility
//

public class MenuItemController : MonoBehaviour {

    Transform cameraTransform;
    float scale = 0.0f;             // Scaling multiplier
    Vector3 goScale;                // Saved values for this gameObjects scale

    bool stateTransitioning = false;        // Item currently transitioning?
    public float appearanceDelay = 0.0f;    // Delay before item starts zooming in
    public float appearanceSpeed = 3.0f;    // Speed at which item scales into place
    public float dismissDelay = 0.0f;       // Delay before item disappears
    public float maxSize = 1.0f;
    public Color highlightBGColour = Color.red; // Text background colour when player focusing on menu item
    Color defaultBGColour;
    float transitionSpeed = 3.0f;
    float transitionDelay = 0.0f;
    float targetScale = 2.0f;
    RenderIf renderIf = null;               // Used to defer renderer enable/disable if renderIf in charge

    void Awake() {
        renderIf = GetComponent<RenderIf>();
        goScale = transform.localScale;             // Remember the configured size
        transform.localScale = Vector3.zero;        // and then shrink menuItem to zero
        defaultBGColour = renderer.material.color;
        renderer.enabled = true;                    // Disabled in inspector to reduce clutter
    }

    // Trigger this MenuItem to transition from visible to invisible or vice versa
    // setActive is subscribed to the menuItemsActivationEvent event which triggers
    // when the player toggles the 'm'enu
    // NOTE: Not using go.SetActive(false) since then the gameObject won't receive sendMessages
    void setActive(bool active) {
        stateTransitioning = true;
        if (active) {
            transitionSpeed = appearanceSpeed;
            transitionDelay = appearanceDelay;
            targetScale = maxSize;
            if (renderIf == null)
                renderer.enabled = true;            // User able to see it appear
        } else {
            transitionSpeed = -appearanceSpeed;
            transitionDelay = dismissDelay;
            targetScale = 0;
        }
    }

    // Triggered when the player looking directly at this menu item
    void beingLookedAt(bool playerLookingAtMe) {
        if (playerLookingAtMe) {
            renderer.material.color = highlightBGColour;
        } else {
            renderer.material.color = defaultBGColour;
        }
    }

    // Per frame update processing, only needed during transitions.
    void Update() {
        if (stateTransitioning) {
            if (transitionDelay <= 0) {
                if (((transitionSpeed > 0) && (scale < targetScale)) ||
                    ((transitionSpeed < 0) && (scale > targetScale))) {
                    scale += Time.deltaTime * transitionSpeed;
                } else {
                    stateTransitioning = false;
                    if ((transitionSpeed < 0) && (renderIf == null)) {
                        renderer.enabled = false;       // Turn object rendering off now it's disappeared
                    }
                    scale = targetScale;
                }
            } else {
                transitionDelay -= Time.deltaTime;
            }

            transform.localScale = new Vector3(goScale.x * scale, goScale.y * scale, goScale.z * scale);
        }
    }
```

```
    // Subscribe to the menuItem activation event and call setActive() when event triggers
    void OnEnable() {
        PlayerController.menuItemsActivationEvent += setActive;
    }

    // Unsubscribe from the event
    void OnDisable() {
        PlayerController.menuItemsActivationEvent -= setActive;
    }
}
```

Figure 48 - Source code - MenuItemController.cs

Appendix C: Use Case Specifications

Use case specifications to match Use Case diagram in Section 4.6

Use-Case Name: **Select Course**	ID: 1	Importance: High
Primary Actor: Player, Game		
Stakeholders and Interests:		
Brief Description: Player wants to change the slalom course		
Trigger: Course menu item selected via 'look to select' mechanism		Type: External
Relationships: Association: Player, Game Include: Extend: Generalisation:		
Normal Flow of Events: 1. Player holds focus on one of the named course menu items (Terrain Park, Val Thorens or Whistler) until the green radial dial animation completes (2 seconds) 2. Game recognises menu choice and changes current course to selected value 3. Game triggers Unity level load of scene called <current course><current difficulty> for instance ValThorensBegineer		
Subflows:		
Alternate/Exceptional Flows:		

Use-Case Name: **Select Difficulty**	ID: 2	Importance: High
Primary Actor: Player, Game		
Stakeholders and Interests:		
Brief Description: Player wants to change the slalom course difficulty		
Trigger: Difficulty menu item selected via 'look to select' mechanism		Type: External
Relationships: Association: Player, Game Include: Extend: Generalisation:		
Normal Flow of Events: 1. Player holds focus on one of the named difficulty menu items (Beginner, Intermediate or Advanced) until the green radial dial animation completes (2 seconds) 2. Game recognises menu choice and changes current difficulty to selected value 3. Game triggers Unity level load of scene called <current course><current difficulty> for instance WhistlerAdvanced		
Subflows:		
Alternate/Exceptional Flows:		

Use-Case Name: **Start Race**	ID: 3	Importance: High
Primary Actor: Player, Game		
Stakeholders and Interests:		
Brief Description: Player wants to start skiing		
Trigger: Start Race menu item selected via 'look to select' mechanism		Type: External

Relationships: Association: Player, Game Include: Extend: Generalisation:
Normal Flow of Events: 1. Player holds focus on the Start Race menu item until the green radial dial animation completes (2 seconds) 2. Game recognises menu choice and changes Player state to 'Count down'
Subflows:
Alternate/Exceptional Flows:

Use-Case Name: **View Highscores**	ID: 4	Importance: High
Primary Actor: Player, Game		
Stakeholders and Interests:		
Brief Description: Player wants to view highscores		
Trigger: View Highscores menu item selected via 'look to select' mechanism		Type: External
Relationships: Association: Player, Game Include: Extend: Generalisation:		
Normal Flow of Events: 1. Player holds focus on the View Highscores menu item until the green radial dial animation completes (2 seconds) 2. Game recognises menu choice and triggers display of high score table		
Subflows:		
Alternate/Exceptional Flows:		

Use-Case Name: **Exit Game**	ID: 5	Importance: High
Primary Actor: Player, Game		
Stakeholders and Interests:		
Brief Description: Player wants to stop the game		
Trigger: Exit Game menu item selected via 'look to select' mechanism		Type: External
Relationships: Association: Player, Game Include: Extend: Generalisation:		
Normal Flow of Events: 1. Player holds focus on the Exit Game menu item until the green radial dial animation completes (2 seconds) 2. Game recognises menu choice and exits		
Subflows:		
Alternate/Exceptional Flows:		

Appendix D: Menus

The original game design menus

Main menu

The main menu screen will have the following options on the PC platform:-

Slalom Ski Challenge – Main Menu			
Start single player game			->
Achievements & High Scores			->
Instructions			->
Starting level	<-	Glorious Green	->
Player 1 controls	<-	Oculus Rift	->
Control options			->
Graphics options			->
Audio options			->
Change Profile			->
Exit			->

The item the player is currently looking at will be highlighted with a bold background and a shaded text, as shown above for the 'Start single player game', if the player wishes to make the selection they should look directly at the arrow ('->') to the right of the menu item text. It will then become highlighted with a 3…2…1 counter to its right. If the player maintains focus on that item then it will be chosen.

The arrows on either side of the starting level allow the player to choose a different level. The arrows will only be bold and active if there is an option in that direction, otherwise they will be faded and inactive.

The arrows on either side of the controller setting allow the player to choose between 'Keyboard', 'Gamepad' and 'Oculus Rift'.

Control options

The 'control options' menu will look like this

Slalom Ski Challenge – control options			
Keyboard			
Steer left	a	or	Left-arrow

Steer right	d	or	Right-arrow
Speed up	w	or	Up-arrow
Back			

The key configuration can be changed by selecting one of the control assignments, either by mouse & left click, or by using cursor keys to navigate to the highlighted assignment & press Enter, at which point the assignment will change to display a '?' with a highlighted surround. The next key press or mouse click then sets that assignment. If the key pressed was previously assigned then the highlighting jumps to that position with a '?' again and the next press sets the value.

Graphics options

The 'graphics options' menu will look like this

Slalom Ski Challenge – graphics options			
Resolution	<-	1920x1080	->
Display mode	<-	Full screen	->
Back			

The other 'display mode' will be 'windowed'.

Audio options

The 'audio options' menu will look like this:-

Slalom Ski Challenge – audio options			
Volume	<-	7	->
Back			

Achievements & High Scores

This is a graphical screen which displays the achievements the player has completed and their best scores & times. Once the player has finished browsing they can choose 'Back' to return to the menu they came from. See the 'Achievements & feedback' section of this document for more details.

Pause Screen

Slalom Ski Challenge – pause screen			
Resume			
Achievements & High Scores			
Instructions			
Volume	<-	7	->
Quit			

Profiles Screen

This menu will allow the player to switch to a different profile. A player profile tracks their progress in game so far as well as their IPD and other Oculus Rift settings.

Slalom Ski Challenge – profiles			
Create new profile			
Change profile	<-	Mark	->
Back			

It is envisioned that a much more graphical approach to this menu could be taken in which the player profiles are represented by 3D player models with name tags. The player would swing their head from left to right bringing the profile they wanted into the centre of their view. Looking directly at the 'Select' icon beneath the 3D model would make the appropriate selection.