IPHONE 3D GAME - MASTERS THESIS

by

Suhayb Mahmood

CREATIVE SOFTWARE SYSTEMS

Suhayb Mahmood, Candidate

Mike Chantler, Advisor, Reader
Roger Rist, Reader

HERIOT WATT UNIVERSITY
2011
This Masters project is aimed at about building an iPhone game that uses shaders to enhance the graphics. The game will record player locations and the length of time spent at these specific areas, whilst focusing on shader locations to see if there is preference from the players. A discussion on the subject of shaders and their importance to the 3D graphics is provided in a literature review. This paper discusses the approach taken to create the game by generating requirements and then finding a framework that fulfills those requirements. An evaluation methodology is designed and a project plan revealed. Finally, the results are presented as maps and the data is statistically analysed to find if there is any significance.

Author: Suhayb Mahmood
Advisor: Mike Chantler
Date: April 13, 2011
Department: Computer Science
Degree: Creative Software Systems
"DECLARATION I, ..................................................................., confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included. Signed: .................................................................
Date: ................................."
Dedication

I would like to dedicate this masters project to my family and my uncle who have supported me every step of the way during my highs and lows. I would like to tribute this to my grandfather who has passed away recently. He was a great role model in my upbringing and taught me the values of life. I would also like to thank my mother who’s always been there for me. And many thanks to my big sister for her continuous encouragement.
Acknowledgement

I would first like to thank my professor, Mike Chantler who has dedicated his busy time to helping me with problems I faced, areas I needed more clarification in and overall giving advice and support. I would like to take a step back here and apologise about barging into your office many times and asking the same question: Sir, are you busy?
Table of Contents

Abstract ii

Dedication vi

Acknowledgments vi

List of Tables xii

List of Figures xiii

1 Introduction 1

1.1 The Vision 1

1.2 Game Changer 2

1.3 The Graphics Processor Unit 2

1.4 The beauty of shaders 3

1.5 Project Goals 4

1.6 What is the game going to be? 5

1.7 The Problem 5

1.8 Conclusion and Analysis 5

2 What is a shader? - Literature Review 7

2.1 Shader Introduction 7

2.2 Photorealism 7

2.3 Vertex and Fragment Shaders 8

2.4 Shader Uses 8

2.5 How to create a Shader 8
9 Design of Level

9.1 Abstract

9.2 Design of the Level

9.3 Purpose of the design of the level

9.4 Conclusion and Analysis

10 Design of Game

10.1 Abstract

10.2 Software Requirements

10.3 Key Software practices in Unity

10.4 Hardware Requirements

10.5 Game Design Flow

10.6 Networking - Time Permitting

10.7 Design of User Interface

10.8 Use Cases

10.9 Interaction Diagram

10.10 State Diagram

10.11 Activity Diagram

10.12 Deployment Diagram

10.13 Conclusion and Analysis

11 Project Iterations

11.1 Abstract

11.2 Simplest Approach First

11.3 Applying Shaders
11.4 Planning the level
11.5 3DS Max Models
11.6 Simple Scripts First
11.7 Collision Detection
11.8 IO Operations
11.9 Timers
11.10 Conclusion and analysis

12 Evaluation

12.1 Abstract
12.2 Evaluation Framework
12.3 Actual Evaluation Session
12.4 Player Responses
12.5 Questionnaire
12.6 Data Structure and Format Recorded from evaluations
12.7 How was Data analyzed and Presented
12.8 Statistical Calculations - Level One Analysis
12.9 Level Two Analysis
12.10 Level One T Test
12.11 Level Two T Test
12.12 Conclusion and Analysis

13 Benefits of Using Unity

13.1 Abstract
13.2 Unity Remote
13.3 Easy Interface
13.4 Choice of Languages
13.5 Built in libraries 72
13.6 Global Variables 72
13.7 Ready to use Functionality 73
13.8 Easy to Create multiple levels 73
13.9 Conclusion and Analysis 73

14 Tackling Problems 74
14.1 Abstract 74
14.2 Problem Solving 74
14.3 Writing Custom Scripts 75
14.4 Tutorials 75
14.5 Access and Availability of Sources 75
14.6 Distraction into other areas 76
14.7 Conclusion and Analysis 76

15 Future Work 78
15.1 Abstract 78
15.2 Complete the current Game 78
15.3 iPad 2 79
15.4 C Sharp Advantages 79
15.5 The App Store 80
15.6 Artistic Touch 80
15.7 Programming Shaders 80
15.8 Conclusion and Analysis 81

16 Conclusion 82
16.1 Abstract 82
16.2 Correct Decisions 82

16.3 The Good and Bad 83

16.4 Uncanny Path? 83

16.5 Findings 84

16.6 Objectives Met 84

16.7 Ground Platform Established 84

16.8 Impact on Field 85

16.9 Thesis Conclusion 85

17 Appendix 1 87

Bibliography 88
List of Tables
List of Figures

2.1 Render Monkey 9
2.2 Diffuse Map 10
2.3 Normal Map 11
2.4 Displacement Map 11
2.5 Specular Map 12

4.1 In-game game design in Unity 16
4.2 Graphics Libraries, Physics Libraries and All round Libraries 17

6.1 Work Breakdown 23
6.2 Work Breakdown 24
6.3 Risk Mitigation Plan 26

9.1 Level One Overview 30
9.2 Level Two Overview 31
9.3 Object with shader applied 32
9.4 Object with no shader applied 32
9.5 Object with shader applied 33
9.6 Object with no shader applied 34

10.1 Traditional game design modules 37
10.2 Chosen device orientation for game 41
10.3 Proposed game controls 41
10.4 Complete Level Use Case 42
10.5 Button Liked Clicked Use Case 43
10.6 Interaction Overview Diagram  
10.7 State Machine Diagram  
10.8 Activity Diagram  
10.9 Deployment Diagram  

11.1 Level One Paper Design  
11.2 Level Two Paper Design  

12.1 Fire without smoke  
12.2 Realistic Reflections  
12.3 Visited player locations in x,y,z format  
12.4 Player liked locations  
12.5 Level one player times in seconds  
12.6 Level two player times in seconds  
12.7 Player Heat Map  
12.8 Player path for level one  
12.9 Level one overview  
12.10 Time distribution for level one shaders  
12.11 Time distribution for level one non shaders  
12.12 Level one time standard error  
12.13 Time distribution for level two shaders  
12.14 Time distribution for level two non shaders  
12.15 Level two time standard error  
12.16 SPSS level one T Test  
12.17 SPSS level two T Test  

13.1 Unity Remote App
13.2 Unity Remote App - Manual IP Entry 70
13.3 Unity Interface 71
15.1 Half Complete Controls 78
Chapter 1
Introduction

The purpose of this master’s project is to build a 3D iPhone game. This game will have unique areas where shader’s will enhance a boring scene [8]. Player’s locations and times around the map are logged and clustered to determine if areas near shader’s are more favourable than non shader areas and if so will indicate that players prefer more graphically appealing areas than others. By using shader’s, it will directly test the graphical processing unit of the iPhone 4 for its computation capability.

1.1 The Vision

The aim to achieve stylistic computer graphics in games has never before been so important. To create the flawless complexions of computer-generated characters has driven artistic ability in the games industry to new heights. With the increasing number of cores in consoles and desktop PCs, the appetite for higher detailed graphics that reflect the accuracy of their real life counterparts has increased. Since the arrival of multi core processors, the power to handle manipulation of large datasets through parallel processing and improved concurrency through threads has provided a much-needed lifeline from the strains of collision detection and physics. The introduction of graphics cards gave a much needed increase in frame rates of games. All these improvements were occupied with consoles and gaming PCs. However, in 2004/2005 the release of the Apple iPhone 3GS revolutionized and changed the gaming market place forever.
1.2 Game Changer

The iPhone 3GS was more faster than its predecessor, iPhone 3G. The iPhone 3G was the device that changed the Smartphone market from phones that made phone calls to phones that functioned as gaming machines, business tools and academic teaching tools. All this software is now all formally known as apps. These small software applications developed by all sorts of different parties demonstrate the vast appeal of small mobile applications to organisations of all types. The first applications were very simple that displayed one view that would show, for example, address books, lists of geographic locations to album slideshows. These applications were appropriate for devices that had minimum of 64MB for memory and roughly 500MHZ processing power. These apps were designed to run on processor light devices. However as time progressed and the more powerful devices arrived, it gave way to augmented reality applications and extreme data processing applications for businesses. The processing power on mobile devices also increased and graphics processor chips were more commonly installed in smart phones. The Power VMBX chip [2] was responsible for the phone 3GS graphics processing.

1.3 The Graphics Processor Unit

This extra chip took away much of the strain the CPU would have had to cope with. These chips contained a pipelined structure where a graphics scene would go through before it would get rendered on the screen. This pipeline included blocks where specific operations could be carried out to transform pixels or colour. These blocks are the fragment shader and the vertex shader. The first graphics processor chips contained fixed pipelines that meant programmers could not implement custom graphics routines. Newer chips have programmable pipelines where custom graphics
code can be executed. This small piece of graphics code is known as a shader. This shader is where the heart of this project is. It is this piece of code that enables high quality realistic graphics to be created. The introduction of shaders meant that glossy material, reflections or smoke could be rendered in real time and be more realistic. This extra processing power directly comes from taking the graphics processing consumption off the CPU and onto the graphics-processing chip. This elevates performance, increases frame rates and overall fidelity of graphics.

1.4 The beauty of shaders

The importance of shaders and their effects in games and on us perceiving beautiful graphics is vital. Next generation games (Sony Playstation 3 HD games) are now judged on storylines, game mechanics and graphics. It is essential to making triple A rated games. When marketing games in magazines or on television, first we visually see the appeal of the game and then we interpret what the advert is describing to us. But it is the initial few seconds looking at the graphics that will determine if we lose interest or not. So employing the use of realistic graphics is good for marketing of games as well as maintaining commercial appeal. The emphasise on displaying games like movies is so important that games studios hire artists to capture key scenematic sequences.

The fact that artists are employed shows that artistic touch in games is desirable. This artistic touch is expressed through how they decide to manipulate the shader to reproduce the results. There is a huge interest in making good graphics, there is a huge importance in shaders, and because of the reasons outlined above I believe examining this area of graphics production is crucial. Looking in depth into shaders has many benefits. When applying high quality graphics to devices that have quad
core processors and GDDR5 RAM chips, its not much of a constraint. However, when considering a mobile phone device at which the memory is fixed at 500MB max and has about 1GHz processing power, how well could that cope? This is important as it will determine how well suited a device is in rendering and what the limits of the device are. In particular, the device in mention is the iPhone 4. This device is selected as it is one of the biggest selling smartphones of all time. Apple’s App store has been hugely successful due to its unique selling point of developing for a desirable device. Apple have also kept their software up to date with continuous releases of IOS updates and addition of newer frameworks. These frameworks access the latest hardware features of the device. With Apple’s continuous focus of branching out into newer markets with the iPhone series, it has been the market leader from the beginning.

1.5 Project Goals

The goal of this project is to create a game that will use shaders and detect if the positioning of shader objects influence players locations and times. The meaning of this experiment is to determine if players prefer game scenes with high fidelity of graphics or not. This is crucial, as it will reinforce the idea introduced above of the importance of good quality graphics and their perception. It also stresses the idea of placing objects that are enhanced by shaders in areas where players are destined to follow as a waypoint to the finish line. It is vital to evaluate this game by allowing test subjects to play this game under different game environments and to see how it impacts them. This data will be collected by logging players locations and the amount of time it takes to completing the level. The data is analyzed by forming appropriate representative maps that demonstrate and reinforce the ideas discussed in this chapter.
The other idea is to test if the iPhone device will have the computing power required to process shader’s. As iPhone 4 contains a graphics processor unit - PowerVR SGX 535 [2] - the graphics processing abilities of the chip can be tested.

1.6 What is the game going to be?

It is a single player game, where the objective is to collect many ammo boxes that will be scattered all over the game level. The games objective is simple so that the projects focus can be concentrated on evaluating the importance of shader’s. The games environment will have areas containing different graphical effects produced by shader’s. One thing to note here is that the game will not contain cinematic scenes as these are very difficult to create where the true expressive presentation is meaningful [13]. The clear downfall of this game could be its simple objective of collecting boxes. Players may feel this is not a fun aim and think its too simplistic. However, if I was going to develop a fun game, there is more difficult aspects to contend with when developing a proper fun game[9].

1.7 The Problem

So the problem is creating shader’s and applying them to objects whilst measuring the player’s location and time response with respect to where the shader’s are.

1.8 Conclusion and Analysis

The project aims are to get a fully functional game implemented and to test if using modern graphics techniques, such as shader’s in this era has any impact on the player’s in game objectives. The other aim of the project is to address if there are extra motivations for using shader’s: Do players find it more appealing? Do they tend to
stay in these areas longer? Is there any point in adding "better looking" graphics into a game? It is also key to test how well equipped modern devices are for processing shaders.
Chapter 2
What is a shader? - Literature Review

2.1 Shader Introduction

A shader is a small program that runs on a graphics processor unit. They render graphical effects in a scene in real time like "bump mapping or shadows" [11]. Other examples of shader outputs are listed below:

- reflections
- illumination
- masking
- bloom
- specularity
- heat haze

2.2 Photorealism

The reason why they are so important is in the area of photorealism. To achieve photo realistic rendering, too many variables need to be considered that cannot be simply expressed in one equation [10]. The problem is to control the flow of information that would be manipulated by some algorithm, i.e. a mapping process. This mapping process can be considered what a shader does. Through this, developers have more control over how data is manipulated and it facilitates in the aim for photorealism.
2.3 Vertex and Fragment Shaders

Shader processing has two distinct paths: vertex shaders and fragment shaders. A Vertex shader transforms the vertices of an object and the fragment shader computes the colour. As graphics processor units have a highly parallel design, there can be thousands of shader instances running in parallel. To put things in perspective, a GeForce FX card can reach 20 Gigaflops which is similar to a 10 GHz PC [11].

One can see the benefit of shaders on a small mobile device like the iPhone 4. Computing Fast Fourier Transforms are expensive on mobile devices [14] and thus the use of shaders can greatly reduce this overhead.

2.4 Shader Uses

Shaders are most widely used in games, but have also had roles in visualising large complex data sets [3]. They also play a part in movie production, noticeably from Pixar. This led to the creation of technologies such as RenderMan, RenderMonkey and programming languages like Open GL Shading Language for creating shader effects. Open GL Shading Language is a C based language for programming fragment and vertex shaders. Render Monkey is an interactive design tool for creating shaders. The screen shot in Figure 2.1 shows Render Monkey.

2.5 How to create a Shader

Critically, if one had to create shaders for games there are only two options. Firstly, if you are very experienced in programming in shaders already, coding might give better performance benefits. Secondly, using an IDE like Render Monkey is a great tool for validating the effects you are creating by previewing them.
Shaders require the use of a 2D texture as an input. The texture is put through an existing shader by setting up some parameters and then by examining the end result [1]. This is an artist’s role. The input into a shader will most commonly be colour maps, bump maps, illumination maps etc. One rendering pass sometimes cannot be enough and thus a composition of maps rendered through shaders can be common for it to work.

2.6 Shader Example

As an example, the following shows the types of image maps that can be used to create shaders. If we have a normal texture like the one shown in Figure 2.2:

This is a diffuse map. Using various Photoshop techniques, a variety of other maps can be produced. The first type of map we can discuss is a normal map, which is
The normal maps purpose is to alter the direction a surface is facing. The next map is a displacement map shown in Figure 2.4:

This map defines the height of the individual pixels. The last map is a specular map shown in Figure 2.5:

The specular map defines the shine and the color of light reflected back. Hence, the brighter the map, the more shiny it looks.

2.7 Conclusion and Analysis

What is missing in the shader literature is the fact they cannot be any ready to use "artistic expression" from shaders. The data still has to be manipulated by programming or by designing special textures to get the artistic presentation. It would be much more beneficial if there existed ready to use shader templates. For
Figure 2.3: Normal Map

Figure 2.4: Displacement Map
example, consider character facial expressions over a period of time. This could be
tackled by programming patterns, but in this case shader patterns that can change the
characters expression accordingly over time. Another example is creating a dark dull
scary scene, again one still needs to create the whole scene themselves. There can be
a ready to use library with general effects that the developer would select and apply.
Render Monkey provides the basic building blocks for common graphical effects using
shaders but does not provide an actual usable library containing templates for game
environments or characters. This would be a good area for further activity.
Chapter 3
Requirements Analysis

The following lists the functional and non functional requirements for the game. This list is not extensive because each item in the list contains sub requirements that also need attending to. So, only the overview is shown.

**Functional Requirements:**

- Shaders

- 3d Characters

- 3d Environment

- Game AI

- Physics

- Collision Detection

- Monitor player locations

- Setting up the server - can use hw.macs

- Networking for players to log in and out

**Non functional requirements:**

- User friendly user interface

- Tutorial to play the game
The hardware requirements are as follows:

- iPhone/iPod/iPad
- Server
- Apple Mac Laptop

3.1 Conclusion and Analysis

The two non-functional requirements listed are the least prioritised in this project. However, it is important to recognise if this game was taken into production for public release, then the two non-functional requirements will be more relevant. But for the purpose of experimentation, the main functional requirements are the key. The networking and server requirements are time permitting and will get discarded if time is limited.
Chapter 4
Selecting the Correct Technology

4.1 Abstract

This chapter describes the two different paths that can be taken to produce games. The various available game libraries are analyzed and there advantages and disadvantages for selecting them for this project also discussed.

4.2 Discussion

There are two main routes that can be taken to create a game aimed at the iPhone: interactive game design or code based programming. The interactive approach is where all game entities can be modelled visually on screen. The code based programming follows the traditional approach of writing code to create the scene and only at compile time can the end result be seen.

These two approaches provide completely different design methodologies. The two also contrast differently in development time and skill level. Here we can contrast the approaches to see which design methodology is more suitable for this project. The interactive approach allows the designer to see at each step of the way what they are creating. The diagram in Figure 4.1 shows Unity - a commercial game development suite that follows the interactive design approach:

As shown, the editing options are available as the designer simultaneously edits and tests the final rendered output. This style of game production follows the trend from packages like 3DS Max or Maya. The clear strength here is the visual changes
are rendered straight away. Although not shown, these packages contain massive amounts of pre built ready to use libraries that house environment textures/ scripts for character movement etc. Most importantly, since everything is readily available, the designer does not need to write as much code as they would have. As this is an integrated development environment, most of the work the designer would have to do would be in this one application. The main disadvantage here is that there is no low level programming involvement that is key to creating fast/slick graphics for games. The designer depends on the package to handle the rendering as efficiently as possible.

The second approach is writing code. This would ultimately involve finding and including graphics libraries. In this approach, not only the graphics library is needed, but if the designer needs functionality for networking or collision detection, they have to find the libraries and include/link them with their project. The graphics library renders the scene and does not include processing collisions or physics. In
this approach you open yourself to a vast choice of libraries that can be used. This approach follows the traditional coding practices in game production.

Some libraries that can be potentially used are listed below:

- Ogre3D
- OpenGL ES
- Oolong Engine
- SI02

Not a comprehensive list but a small snapshot that represents what is available on the market as of now.

Looking closely at the above options, Ogre 3D and OpenGLES are purely graphics engines while the other two libraries provide other utilities such as physics and collision detection etc. Figure 10.1 shows the packages above categorised into types they function as.

![Figure 4.2: Graphics Libraries, Physics Libraries and All round Libraries](image-url)
The purely graphics libraries are at a severe disadvantage as they do require the user to still search for other libraries for physics and networking. Even after the programmer finds these libraries they still have to link them and implement code to function with them. However, since the open source libraries are free and therefore cost saving is beneficial.

In contrast, the inner circle shows libraries that provide collision detection, networking, etc. The Oolong Engine is an open source library with many features that are required to create a game and most importantly its free. However, its documentation to use the library is extremely poor as was its forum activity. The SI02 [12] library is a commercial library that does fulfil the requirements of this project. However, the library costs roughly two hundred pounds and it looked very difficult to use, as its programming interface was complex.

The disadvantage in the coding approach is that there is no clear development suite where everything happens. For example, if a designer wants to create a character, they would have to use another 3rd party modelling package. Once they have modelled the character, only then can they export and import it into the graphics library for display. Another clear disadvantage is that programming without prior experience of using these libraries can be extremely difficult.

Comparing the two approaches reveals two different styles of game production. Admittedly, the interactive game production technique is a recent development in game production. A single developer looking to make a quick game, would be best suited to using the interactive approach as it scales better for development time and does not need the backing of a game studio. However, the commercial cost for these development suites are expensive.
However, if you have reasonable skill in creating 3D characters and 3D environments and have previous experience in game development, the coding approach will be more familiar and it will follow your natural software development cycle.

As the project timeline is limited and an evaluation period is required with users, the Unity option is most attractive as it matches the functional game requirements. Its an interactive and integrated environment that presents one programming interface. The cost here is a factor but is balanced out by the fact that Unity is integrated with all the gaming functionality for this project.

4.3 Conclusion and Analysis

Having looked at all available options for creating games and examining there individual benefits, the best solution reveals itself. The problem with standalone frameworks is compatibility and the trouble it will give in linking to other frameworks. The best solution in this case was Unity as everything is readily available in one package and is easily accessible. There is no point at looking into libraries like OpenGLES and Ogre3D unless you are creating graphics without physics.
Chapter 5
Proposed Testing and Evaluation Plan

5.1 Abstract

This chapter raises issues that have been picked up from the literature review and what it implies for the masters project.

5.2 Discussion

The aim of this project is to identify if players notice the areas where shader’s are located and if they go and stay in these areas the for a long period of time. That is the main purpose of this project: the influence of shader’s on players.

Firstly I propose my hypothesis: "If player times and locations are related to shader regions, then changing the shader regions will result in changes in player times and locations".

The players times and locations are the dependent variables - what I am measuring. The shader locations as the independent variable - what I will change during experimentation to see if it affects the dependent variables.

The worst case scenario is the null hypothesis: "Changing shader regions does not affect player times and locations; hence the two are not related”. To test the hypothesis I can design an experiment as a within groups experimental design.

Firstly, the experiment will involve me adding or changing the positions of the shader’s locations, this can be done by creating another game level but instead po-
sitioning the shader’s at different locations. So essentially there would be two game
levels with differing locations where shaders will be placed. I plan to make one game
level where all shaders are located at one region and the second level where shaders
will be mixed in with non shaders. My reason for this set up is because level one will
show if there is a preference shown from players and it will be further validated if in
level two, the players notice shader objects and go to these regions again.

The quantitative data type of the players locations and times are represented by
ratio - data measured from an interval scale and the scale has a zero point. This factor
is important as it distinguishes what types of statistical tests that can be conducted.

The game will have to write updated player locations from the game to a text file.
When all data is gathered for one play session, the data points will be clustered [6].
We can cluster the data in scatter plots to identify natural clusters or use K-Means
clustering where K will be the number of shader’s in the game level. I also propose to
use heat maps to show liked locations from players. Player times will be gathered and
averaged for shader areas and non shader areas. When the 2 means are computed, a
t test will reveal if there is a statistical significance in the timing data.

5.3 Conclusion and Analysis

This chapter proposes ideas and methods that I can further research into for evalu-
ating the results I gather. I have given a brief overview of the process I would like
to take in analyzing the data. To propose the analysis plan before the design of the
system is almost identical to test driven development where tests are first written
then the code. The advantage in this method is the fact that you have a clear plan of
how you would like the results to be examined (test case) and its a case of producing
the game to give these results (writing the code).
Chapter 6

Proposed Project Plan

Rational Unified Process is the development process that will drive this project. It is most suitable because its linear approach to project management allows you to plan your next step. This is crucial here because without game characters, levels, collisions, etc we can’t progress in the development.

6.1 Work Breakdown Structure

The proposed work breakdown structure in Figure 6.1

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Duration (days)</th>
<th>Task dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Library set up</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Getting used to the library</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Character design</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Level design</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Importing character into game</td>
<td>3</td>
<td>1,2</td>
</tr>
<tr>
<td>6</td>
<td>Collision detection/Game scoring</td>
<td>7</td>
<td>1,2,5</td>
</tr>
<tr>
<td>7</td>
<td>Main game loop/Network connectivity</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Server set up</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>User log in/out system</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Shader implementations</td>
<td>14</td>
<td>1,2</td>
</tr>
<tr>
<td>11</td>
<td>Before release testing</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>User testing</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Testing analysis</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Report write up</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.1: Work Breakdown
A key point to remember is that networking and server set up is time permitting and depends on the level of difficulty to get it working. If there is no time, these two requirements will be discarded.

6.2 Gannt Chart

Figure 6.2 shows a Gantt chart for the project showing some tasks that can be executed in parallel:

![Gantt Chart for Project](image)

Figure 6.2: Work Breakdown

6.3 Risk Management

The risks in the project mainly come from poorly gathered data from evaluation sessions or failure to get data written to text files. The main risks are identified in
Figure 6.3.
<table>
<thead>
<tr>
<th>Risk</th>
<th>Importance</th>
<th>Likelihood</th>
<th>Risk Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exporting character problems</td>
<td>High</td>
<td>Low</td>
<td>Try another exporting format that is less optimized</td>
</tr>
<tr>
<td>Shaders are hard to create</td>
<td>High</td>
<td>Medium</td>
<td>Use a package like Render Monkey or alike</td>
</tr>
<tr>
<td>User Interface poorly designed</td>
<td>Low</td>
<td>Low</td>
<td>Quick and simple redesign of interface / Reduce amount of controls</td>
</tr>
<tr>
<td>Networking Problems</td>
<td>High</td>
<td>High/Medium</td>
<td>Don’t implement the requirement</td>
</tr>
<tr>
<td>Not able to write player locations to text files</td>
<td>High</td>
<td>Medium</td>
<td>Print values onto standard console</td>
</tr>
<tr>
<td>Data gathered is not sufficient</td>
<td>High</td>
<td>Low</td>
<td>Use standard error to see if samples in hand are representative of the population</td>
</tr>
<tr>
<td>iPhone can’t handle running the game</td>
<td>High</td>
<td>High</td>
<td>Use Unity’s simulator on Laptop</td>
</tr>
</tbody>
</table>

Figure 6.3: Risk Mitigation Plan
Chapter 7
Professional, ethical and legal Issues

Players locations will be stored from the game. This does not need to be approved from players as it not in any way intrusive into their personal lives or impacting them emotionally or financially. Hence, I do not see any ethical issues arising from this game.
Chapter 8
Implications for the Masters Project

8.1 Abstract

This chapter raises issues that have been picked up from the literature review and what it implies for the masters project.

8.2 Time Permitting Requirements

During the course of requirements gathering, some requirements, such as networking and setting up a server are not crucial. If they are not implemented in the final game, it meant that I did not have sufficient time left.

8.3 Chosen Design Process

Through the discussion of this paper, it’s been decided that the interactive game designing approach is most suitable. Its implications are there is no need to worry about fixing programming bugs and therefore can fully focus on the task of designing the game.

8.4 Past Skills

Unity accepts 3DS Max format graphics. This is a huge benefit as I can employ my skills learnt earlier in the year from 3D Animation course. It implies I can cut down on learning time and spend more time reducing polygon counts in order to reduce computational overhead.
8.5 Two Step Process

This project can be thought of a two step process. The first step is creating the game and getting it working. The second stage is clustering player locations and generating heat maps. This implies that there can be no delays in the project as the evaluation period is crucial to the outcome of this project.

8.6 Conclusion

In conclusion, the project places a strong demand in development and evaluation. The displaying of a shader object in the game and observing their effects on players indicates the importance of them as an artistic form of expression and not merely as attractive graphics. This will also prove that game designers can emphases certain locations in games where they want players to explore more via the use of shaders.
Chapter 9
Design of Level

9.1 Abstract

This chapter describes the creation of the game levels and its motifs. It also details the differences in shader object and non shader objects in the game.

9.2 Design of the Level

Figure 9.1 and Figure 9.2 below show the final produced levels:

![Figure 9.1: Level One Overview](image)

The first image shows a birds eye view of what level ones structure is. The camera in the center of the picture is where the player initially starts. The little sparkling suns are the positions of the lights. The small red boxes are the ammo pick up boxes. There are also two roads. However, the most important aspect of this scene is the buildings encircling the level. The left side of the level the buildings have an applied shader whereas the buildings on the right hand side are all plain textures. The screenshot below shows one of the buildings with a shader applied revealing a
Figure 9.2: Level Two Overview
certain bumpiness.

Figure 9.3: Object with shader applied

With this same building in question, its counterpart on the right hand side of the level is extremely plain:

Figure 9.4: Object with no shader applied

Since the level is balanced with non shader objects to shader objects, it brings to mind the question of where the player spends their time? and what areas attract
their attention more.

For the second level, a similar pattern follows but this time the shader objects are mixed around the level with non shader objects:

![Image of object with shader applied](image)

Figure 9.5: Object with shader applied

Its non shader counter part is shown in Figure 9.6.

Looking at the above pictures, the question arises, what are the white platforms for? The white platforms are colliders that alert the main frame listener that we are near location of a shader object and to start the timer. Soon as the player leaves this area, the timer is stopped and the time spent in this area is stored appropriately. Times are recorded for shader and non shader areas. The time roaming around the level is total time taken to complete the level subtracted by the time spent in the shader and non-shader areas.
9.3 Purpose of the design of the level

The aim is to measure how long a player spends in each type of area. Two levels have been designed to test this. One level has the shader and non shader objects opposite each other and the second level is a scattered mix. The first level tests if one specific region of a game can have the player prolong his game play time while the second is to test if the players attention can be grabbed by the shaders.

9.4 Conclusion and Analysis

Having the levels designed in such way that it reinforces what I am trying to measure installs confidence in my goals and the way I go about evaluating it. Having the level set up to aid my evaluation (colliders to start timers, etc) helps to cut down the amount of information I have to physically keep track of during the evaluation.
sessions. One aspect that I am not measuring is how light enhances the shader objects. If its a dark level, would the player even notice the shader object? This can be dealt with in a future project.
Chapter 10
Design of Game

10.1 Abstract

This chapter outlines the design of the software for the game. This is from initial concepts to UML modelling software describing components of the game. A set of use cases show the prescribed steps and actions to complete a task. A user interface design is shown to highlight the placement of user controls. A state diagram shows the various states the game goes through. In conclusion, this chapter lays out a proposed software plan and model for creating the game.

10.2 Software Requirements

From the requirements analysis chapter, we can identify a couple of requirements that input directly into the software creation process, collision detection, physics and monitoring player locations. These three requirements are project critical in that we need them functioning otherwise there cannot be any evaluation procedure. Networking and setting up a server were also identified as requirements but are not project critical and can be ignored for this chapter.

10.3 Key Software practices in Unity

The usual way of creating a game is to separate out the major modules and define a main function that calls upon the smaller modules to complete one render of a frame. This traditional set up is shown in Figure 10.1.

Unity provides a programming environment where essentially a frame listener
Figure 10.1: Traditional game design modules

pattern exists. This is when a class is called which then conforms to this protocol and its procedural methods are called to achieve the game's objective. To conform to this protocol, the function Update() has to be overridden and any class that implements this is called at every frame render. The programming language that I have chosen to script in is JavaScript. Unity’s project files can also be synced with Mono Develop for coding support such as code hinting and debugging.

The standard practice in Unity for getting game objects to perform their in-game function is to create a JavaScript file with the relevant code inside and attach it to the corresponding game object. At runtime, this code will execute and its output can be observed on the object. These scripts can also perform networking calls or calculating collision detection etc.
The design of this game extensively makes use of the Update function to execute code in every frame. One of the great aspects of creating games in Unity is that it does not require any huge amount of object orientated design. The output wanted can simply be done by writing the code to do that directly. This demonstrates Unity's flexibility in allowing game programmers to design the solution that best fits their needs.

Unity recommends keeping physics detection code in the one script. As collision detection is CPU intensive, it's best to perform all the collision detection in one script once every frame. As one does write code, it can be tempting to put collision detection without thought in two scripts and later on to discover the frame rate to drop. This is due to two scripts implementing collision detection code and both being called every frame. Having understood this key practice, I also kept all my collision code in one script.

A similar idea follows for networking code. Sending and receiving player updates more than once in one frame is overkill and thus the same rules apply from above.

10.4 Hardware Requirements

As it has been mentioned, this project will make use of an iPhone 4 that will have sufficient memory and processing power to deal with two small levels with low polygon counts. If the iPhone has trouble in playing the game with low frame rate, a back up plan is to use the Unity simulator on the laptop and transmit player movement from the iPhone device remotely. In effect, the iPhone 4 will be acting as an input device to the simulator that will control player movement.
At time of writing, iPad 2 has just been released and if I was targeting this device for experimentation, the design and planning of the game would have to be different. The iPad 2 has more processing power and has a bigger screen size. These two factors would themselves propose a different game where game controls would be located differently on the screen and the game level design itself would showcase the shaders with consideration of screen size. Although desirable, its not suitable to go back and plan at this stage of the process. However, its worth mentioning here as its relevant.

10.5 Game Design Flow

The first step in getting any game developed is initial sketches and forming a picture of how the game level should be formed in order to test the hypothesis. This is then designed in a 3D modelling package like 3DS Max. Since we are modelling for a mobile device, the polygon count on every object needs to be kept to a minimum. The game I am designing does not deal with complexity of 3D models but instead the textures are put on the 3D model, therefore simple 3D models can be used.

When the 3D models are complete in 3Ds Max, they are not textured there but instead exported as .FBX files into Unity and will be textured in Unity with shaders. Before the shaders can be applied, suitable image maps in the form of PNGs need to be located. Before these images can be imported into Unity and applied as shaders on objects, they first must be converted into displacement, normal and specular maps using any image editing software (Gimp, Photoshop, etc). There are lots of tutorials on Youtube demonstrating how to generate these maps from a normal image. Once these maps are created, they are imported into Unity.

To texture a 3D model in Unity, a material object is made and its type is set to
a specific shader (Bumped Shader, Parallax Specular, etc). Once this is selected, you can drag and drop your normal and specular maps onto slots and Unity will generate the completed shader. Once done, this shader material can finally be applied to the 3D model.

The last step in game design is positioning the objects that are textured by shaders in certain locations that are specific to my evaluation.

Finally programming the games engine will lead to a fully completed game.

10.6 Networking - Time Permitting

For allowing the players to play against each other [5], the ability to update connected players iPhone's with player locations can be achieved through three known network topologies:

- Broadcast - simplifies game discovery
- Peer to Peer - direct communication between all players
- Client-server - server must maintain a copy of game state

There are disadvantages associated through all of the above. Broadcast can lead to more processing for the iPhone, Peer to Peer scales as $n \times n$ and client server will have implementation issues. However, the client server option is the best option as it only requires one open connection. The fact that the server will already host the player locations text file; it makes this option more feasible.

10.7 Design of User Interface

iPhone games downloaded from the App store are in landscape orientation. The controls on these games are located at the bottom left and right hand corners [7]. I
will follow these practices for my game as it is the familiar layout for iPhone gamers and it gives good aspect ratio for the game on screen, like shown in Figure 10.2.

![Figure 10.2: Chosen device orientation for game](image)

From this initial user interface concept diagram, the two most important layouts are the movement control and camera control. These locations have been chosen as they are natural for receiving input from players thumbs. The ammo bar at the top will act like a health bar but instead for showing how much more ammo the player

![Figure 10.3: Proposed game controls](image)
needs to collect before its 100% full. The top right hand side can be used as a button for firing a weapon or some other use if required.

10.8 Use Cases

The use case below will explicitly shows the main scenario the player will be required to perform.

This main process of user collecting ammo that will ultimately allow them to complete the level is been used to provide some sort of game objective.

The use case in Figure 10.5 below shows the requirement on the player to provide input on his preferred areas.

The diagram highlights the need for the user to express his interest in areas that have grabbed their attention. As soon as the user taps the button called "like", the values of their corresponding x, y and a 100 will be saved to a buffer. When the user completes the level, a script will run internally in Unity to write out the values in the buffer to a text file for clustering purposes.
10.9 Interaction Diagram

The diagram below shows the main objects that will be interacting with each other to achieve the desired game goals.

As shown by the arrow and corresponding number, revealing the order of events as they occur. One can also note that as the player is collecting ammo boxes, the player box is also sending Collider Hit event to the ammo boxes box for detecting
collisions between them.

### 10.10 State Diagram

The game goes through various states that are crucial to the process of completing the game and for gathering data for testing.

![State Machine Diagram](image)

**Figure 10.7: State Machine Diagram**

It can be noted that the if conditions are very important between the transitioning states as they do judge when the next level load is initiated. When the user does press the “like” button, it does not cause any transitions as the like feature is for current level.

### 10.11 Activity Diagram

The diagram in Figure 10.8 is essentially same as the state machine diagram, but shows some concurrency. The fork operation shows that as the game is running, values of player are being written to a buffer. Hence, there is some small degree of parallelisation.
Figure 10.8: Activity Diagram
10.12 Deployment Diagram

For testing this game, we can deploy the game onto the iPhone itself and check if it is capable of coping with strain the from the shaders. If it can't, the diagram in Figure 10.9 outlines another set up that can be used:

![Deployment Diagram](image)

**Figure 10.9: Deployment Diagram**

The setup above shows the iPhone being used as an input device and its input transported via Internet connection to a computer running Unity simulator. Here the players movement is mapped as it would be if it were running on the iPhone. This set up will be applied as a contingency plan if any problems occur running the game directly on the device.

10.13 Conclusion and Analysis

In this chapter I have explained my proposed solution in tackling the creation of this game. I believe that the use of UML modelling in this chapter has shown a clearer
insight in design and implementation of various components needed to create the game.

Using the deployment diagrams approach is best way forward as all the processing of storing values into arrays and the writing of those values into files is a computationally heavy process and will be more suited to the computer. This leaves the iPhone 4 to just process user inputs and send them via the network. One also needs to remember processing physics and collision detection on the iPhone is also very computationally heavy and passing this load to another more powerful system is more sensible.

I have chosen a screen layout and design for user controls that is critically suited to first persons shooters on the iPhone (GunBros on App Store, etc). Choosing a standard layout that players know how to use is beneficial and shows I am complying to industry standards. The UML diagrams display the flow of the system and clearly set a development path. This also means I won’t deviate from the plan because the next stage is set out in the diagrams. The flow of the system is objectively stating there will be no deviation from the path outlined from the above diagrams. This is good practice as there is a clear project plan and I know what I need to implement in terms of software and I can allocate sufficient time and resources to do that.
Chapter 11

Project Iterations

11.1 Abstract

This chapter describes the way in which the project was tackled and the subsequent steps at each stage to progress the development.

11.2 Simplest Approach First

My first attempt in mastering Unity was exploring the interface. I then started to draw simple cubes on 3D virtual space. After establishing the draw procedure, I attempted to apply textures to the objects. After learning to successfully apply simple textures I started to research into applying shaders to objects using image maps.

11.3 Applying Shaders

After researching into the different shaders that are available and the uses they come under (discussed in the what is a shader chapter) I was then able to start using them. My first attempt in applying a shader came from inquisition by giving the in built Unity shader the image maps that it needed to create the fully functioning shader material. This shader material was the material that I applied to the 3D cube. After a couple of iterations of applying shaders on objects, I was in an excellent position to start building the complete level.
11.4 Planning the level

Firstly I had to plan out what I wanted the level to look like. This was a crucial step in projects goals and aims as it directly affected what I was trying to measure. I first decided that the best way to test how people reacted to high fidelity graphics was to create levels where there was a clear separation within regions of the level. I decided to create a level where one region contained all the shaders and the other region was all non shaders. Then I decided to create another level that would mix shaders with non shaders to see if players can detect changes in a progressive game environment. I decided to plan out both of my levels on paper and use that as a guideline to what I was going to do. This was most useful as of now looking back with hindsight. Both of the paper level designs are shown below in Figure 11.1 and Figure 11.2.

11.5 3DS Max Models

Building a complete level is not a simple process of creating boxes. There needs to be more complex shapes such as trees, cars, etc. So I had to turn to a 3D modeling package like 3DS Max for complex model creation. I spent roughly a week in 3DS Max in producing simple models and retrieving models that were readily available in 3DS Max. After producing my list of models, I exported them out of 3DS Max in the .FBX format (optimized for Unity) and imported the models into Unity. When I had all of my levels models imported, I positioned objects where they needed to be. At this point in time I basically had a static screen where nothing happened and all objects were stationary. This required me to start producing scripts to add some realism, physics and animations to the game.
Figure 11.1: Level One Paper Design
Figure 11.2: Level Two Paper Design
11.6 Simple Scripts First

My first task was to understand what programming language Unity used for its programming. Surprisingly, it used JavaScript and or C Sharp for its scripting. The developer can choose based on his preference. The Unity reference library for scripting is available online and this is the library I used to learn the functions I can call to achieve my in game aims. I opted for JavaScript as it was more maintained than C Sharp in Unity. However, I did lose out on many .NET services (which could be crucial for future development) but at time of development I went for the language that the Unity reference library provided as it had more up to date scripts. Again, I created simple JavaScript’s and attached them to objects. With the passage of time I got more experienced and was able to write more functional scripts.

11.7 Collision Detection

When I became proficient at creating more functional scripts, it was time for me to start programming to detect in game collisions and its responses. This involved using a collider function as the callback to perform all manipulations for colliding objects. I followed recommended standards to keep all collision detection code in one script as its computational more efficient. For the entire game, I only had to focus on detecting collisions between the player character and the ammo pick up boxes. Later on in development, I had to detect collisions with invisible collider boxes to start timers.

11.8 IO Operations

It was identified very early on in planning that I needed some sort of mechanism to write out the data to text files. This included data on player locations as they moved
about the levels and recording the places that they liked. Again, a script was created that stored all of this data into buffers. When the one level was complete, the buffers were written to text files in the comma separated format. This I/O operation was carried out at the end of levels as its computationally heavy which meant the frame rate would not suffer.

11.9 Timers

As it was mentioned above, invisible collision boxes were put in the game for starting timers when collisions were detected in specific regions. These timers were used for testing purposes and to get timing data on how long players spent in shader to non shader areas. The programming of starting and stopping timers were executed in scripts.

11.10 Conclusion and analysis

I believe that all programming that I did was completed in the time scale that I allocated in my project plan. If I had time to redo the programming, I would redesign the code using C Sharp and make use of its .Net services. This would open up a lot of other possibilities like communicating with servers and web services. By using this extra functionality, we can build a game where it syncs updates from a server and thus we have a continuing developing game without having to release newer versions. This is how triple A rated games for the Playstation 3 operate and also certain successful iPhone games like Angry birds and Frisbee. However I feel that planning out the levels before creating them provided a road map to successful level completion for me.
Chapter 12
Evaluation

12.1 Abstract

This chapter describes the evaluation procedure with end users and provides analysis of the collected data. First, the steps in evaluating the users is outlined and the procedure I went though to record the data. Furthermore, a full analysis of the data in the form of charts and statistics is provided to prove the hypothesis.

12.2 Evaluation Framework

To perform the testing, I had to define a clear model of what I was trying to measure and why I was measuring that quantity. The goal of this experiment is to find evidence of players noticing high fidelity graphics and detecting if their locations and timings change in response to this. The question here is if objects that have shaders as their material, do they attract more players? To test this, I will use an evaluation methodology where I will be observing users playing the game and recording details. A practical issue here is if I can gather all the details that are significant and I may favour some details over others. Hence, there is an element of biasses. Since the data I will be collecting is location based, any real life mapping process like heat maps or cluster maps will be used where appropriate. I will present the data in the form of maps. To interpret it, i will use statistical methods like standard deviation and t test to see if the data has any sort of significance.
12.3 Actual Evaluation Session

The testing session with users was carried out in the ground floor computer labs. Fifteen participants participated. This involved me explaining to the users the controls to play the game and how to complete the level. I also mentioned that if they see a texture on an object that is more 3D in nature, press the like button. As the players tested the game, I used a stop clock to measure how long they stayed in shader vs non shader areas. This was a within groups experimental design where the player had to test the level under two different conditions. One level had all shader objects located only on one side. The other level had shader objects scattered all around the level and they were mixed in with non shader objects.

During the testing I also noted any responses from testers on their observations. If they complain that certain aspect of the game is not functioning very well, I will take a note of it. When the player finishes the game, I will ask if they noticed certain objects that were more visually appealing and if it had an impact on them. Or did they notice something odd in the level that caught their attention? One can say its a post evaluation brief with questions.

12.4 Player Responses

During the evaluation sessions I did get a chance to write down responses to the game. One of the complaints was on the category of poor controls. Three participants said that the touch screen joystick was rigid and difficult at times to control. Some also said that the game was slow. The lack of game speed could have been down to the WIFI connection on the day as the iPhone was setup as shown in the deployment diagram in design of game chapter.

A interesting response back was that the fire in the game level one was not
realistic enough. The fire is shown in Figure 12.1 below:

![Fire without smoke](image)

**Figure 12.1: Fire without smoke**

Specifically, they said that there was no smoke from the fire and it reduced the games realism. This is nothing to do with shaders but does raise an interesting point of how natural functioning real world objects should behave exactly the same as in the game world.

There were much responses on the differences of light reflection on certain objects than others. The certain objects were the objects with applied shaders. As an example, take the screenshot shown below in Figure 12.2:

The right hand rectangle has no shader applied. The little creme sphere in the middle has a shader applied. The difference is that the sphere has specularity and correct light reflections at grooves while the rectangle is solid and plain with hardly any light interaction.

Players also reported that as they moved closer to the sphere, the light reflected changed with respect to distance. Players said that the reflections made the
game more realistic. So one can draw the conclusion that shaders do have some psychological impact in bridging real world gaps in games. Even though the response are not being evaluated statistically, they do carry the meaning and message through that they are significant in the game.

12.5 Questionnaire

One of the items I did not get time to prepare for was creating a questionnaire for my testers for the evaluation session. This questionnaires purpose would have been to probe the players on certain psychological issues. An example question would have been the following: Given two areas, which one do you prefer? The options would have been not preferred, didnt notice and much preferred. The answer to this question would be analyzed by converting the response into interval representation and processed in SPSS. Another type of question to ask could have been if they
felt immersed in certain regions of the game? The data could have provided hidden insight that cannot be logged through gameplay alone.

12.6 Data Structure and Format Recorded from evaluations

The data that was gathered from the evaluation proceedings was stored in text files in comma separated format (widely accepted format for reading in data files). For tracking the players locations in the game, Figure 12.3 shows the outputted format.

![Figure 12.3: Visited player locations in x,y,z format](image)

The repetition of numbers is not a problem, as it represents the slow speed of the player in moving from one location to another, hence the same location was recorded multiple times. This data will be used to create our player track map. Another output in the text file format is the locations that the user liked shown in Figure 12.4.

The first and second column are same as above but the third column is the value expressing that the user liked that location (it can be any arbitrary value, i have chosen 100 here so it shows strong red color in Matlab for heat maps). The last piece of data that was collected was user timings. This was all input into excel as
Figure 12.4: Player liked locations

shown in Figure 12.6.

![Table]

<table>
<thead>
<tr>
<th>Player</th>
<th>Roaming</th>
<th>Shader</th>
<th>Non Shader</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>10</td>
<td>7</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>59</td>
<td>15</td>
<td>6</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>68</td>
<td>20</td>
<td>2</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>43</td>
<td>7</td>
<td>7</td>
<td>57</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>30</td>
<td>20</td>
<td>150</td>
</tr>
<tr>
<td>6</td>
<td>68</td>
<td>25</td>
<td>10</td>
<td>103</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>5</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>37</td>
<td>5</td>
<td>13</td>
<td>55</td>
</tr>
<tr>
<td>9</td>
<td>49</td>
<td>7</td>
<td>4</td>
<td>60</td>
</tr>
<tr>
<td>10</td>
<td>53</td>
<td>9</td>
<td>6</td>
<td>68</td>
</tr>
<tr>
<td>11</td>
<td>49</td>
<td>12</td>
<td>12</td>
<td>73</td>
</tr>
<tr>
<td>12</td>
<td>60</td>
<td>15</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td>13</td>
<td>36</td>
<td>15</td>
<td>9</td>
<td>60</td>
</tr>
<tr>
<td>14</td>
<td>47</td>
<td>13</td>
<td>5</td>
<td>65</td>
</tr>
<tr>
<td>15</td>
<td>46</td>
<td>20</td>
<td>4</td>
<td>70</td>
</tr>
</tbody>
</table>

Figure 12.5: Level one player times in seconds

The first column represents the instance of a player. The second column was the time spent (in seconds) wandering about the level. Third and forth columns are the time spent in the shader and non shader areas. The fifth column is the total time taken to complete the level. This data will be used for calculating statistically if our experimentation was instrumental.

12.7 How was Data analyzed and Presented

The data that was stored in text files was analyzed in Matlab as its more quicker. Also it was convenient to write an automated Matlab script to read the data file and output the data in the required chart format. For the liked locations, heat maps were produced that would showcase which locations the player liked. An example is shown in Figure 12.7.
Figure 12.6: Level two player times in seconds

<table>
<thead>
<tr>
<th>Player</th>
<th>Roaming</th>
<th>Shader</th>
<th>Non Shader</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>53</td>
<td>13</td>
<td>24</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>15</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>23</td>
<td>6</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>94</td>
<td>19</td>
<td>7</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>82</td>
<td>14</td>
<td>9</td>
<td>105</td>
</tr>
<tr>
<td>6</td>
<td>90</td>
<td>14</td>
<td>5</td>
<td>109</td>
</tr>
<tr>
<td>7</td>
<td>66</td>
<td>25</td>
<td>6</td>
<td>97</td>
</tr>
<tr>
<td>8</td>
<td>96</td>
<td>31</td>
<td>8</td>
<td>135</td>
</tr>
<tr>
<td>9</td>
<td>54</td>
<td>26</td>
<td>9</td>
<td>89</td>
</tr>
<tr>
<td>10</td>
<td>40</td>
<td>28</td>
<td>7</td>
<td>75</td>
</tr>
<tr>
<td>11</td>
<td>102</td>
<td>30</td>
<td>8</td>
<td>140</td>
</tr>
<tr>
<td>12</td>
<td>34</td>
<td>25</td>
<td>9</td>
<td>68</td>
</tr>
<tr>
<td>13</td>
<td>53</td>
<td>13</td>
<td>8</td>
<td>74</td>
</tr>
<tr>
<td>14</td>
<td>41</td>
<td>18</td>
<td>4</td>
<td>63</td>
</tr>
<tr>
<td>15</td>
<td>72</td>
<td>19</td>
<td>6</td>
<td>97</td>
</tr>
</tbody>
</table>

Figure 12.7: Player Heat Map
The orange rectangle shows were the shader objects were located for level one and the green rectangle is the location where non shader objects were located. As you can see, there are red squares and these are the locations the player pressed the like button. From Figure 12.7, it can be noted the player in this instance did notice the shaders because the red dots represent their acknowledgement.

For the player locations, they were charted using a cluster plot as shown in Figure 12.8.

Figure 12.8 shows the way and direction the player went in level one. If one compares this map to the level one overview, shown in Figure 12.9.

By super imposing the two images, you can see that the player traversed the whole region of the shader section beside the buildings (shader buildings on left hand side of map) and ignored the non shader buildings. It proves the point that a player does get influenced by the surrounding setting.

All of the maps were analyzed in this manner and a full set of the maps can be found in the appendices. By looking at the cluster maps and heat maps, most of the
participants tended to show a interest in shader areas more than non shader areas.

12.8 Statistical Calculations - Level One Analysis

Figure 12.10 shows the time distribution for level one shaders for the 15 players tested.

As you can see, the distribution is spread at the beginning but there is more structure later on. The shader mean from this distribution is 13.9. The standard deviation is 7.4. The points are spread out, giving the large standard distribution. A similar story follows for the non shader sample shown in Figure 12.11. The non
shader mean is 8.3 and standard deviation is 4.9.

Figure 12.11: Time distribution for level one non shaders

The Figure 12.12 below shows the standard error as error bars for level one times. The standard error shows how representative the samples in question are of the population. In this case the shader sample has a small standard error relative to the mean, 13.7%. This implies that the shader sample mean in this case 14, is an accurate reflection of the entire population. However, there is still a 1 in 10 chance that it could not be, according to 13.7%.

A similar analysis can be said for the non shader group, where the standard error is small and is relatively small compared to the mean of the non shader (15%). So once again, the non shader mean is relatively accurate reflection of the entire population.

12.9 Level Two Analysis

Figure 12.13 and Figure 12.14 below shows the distributions for level two shaders and non shaders for the 15 players tested.
Figure 12.12: Level one time standard error

Figure 12.13: Time distribution for level two shaders
The mean for shader in Figure 12.13 is 20.9 and standard deviation is 6.3.

The chart in Figure 12.15 shows the standard error as error bars for level two times.

Figure 12.15 standard error is small again so this sample is an accurate reflection of entire population. The non shader sample is small as well, almost the same as level ones non shader sample, 15%!

12.10 Level One T Test

In SPSS, a paired samples T Test was carried out on the data collected on times for level one. The following diagram shows the computed results from SPSS.

The t column is our T Test value result. The sig column tells if the results carry a significance and the value contained in it (0.019) is also known as the P-Value. As a rule of thumb, this P-Value should be less than recommended alpha value of 0.05. For our results, 0.019 is less than 0.05, this shows that the mean difference between players
Figure 12.15: Level two time standard error

Figure 12.16: SPSS level one T Test
spending more time in shader areas to non shader areas is statistically different and we can discard the null hypothesis. We can already make a judgement that players prefer to spend more time in shader areas.

12.11 Level Two T Test

The similar process of getting the t value from SPSS was done for the level two time data as well. The results that were returned were as follows:

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Std Deviation</th>
<th>Std Error Mean</th>
<th>95% Confidence Interval of the Difference</th>
<th>t</th>
<th>df</th>
<th>Sig (2-tailed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1 shader - nonShader</td>
<td>11.600</td>
<td>9.807</td>
<td>2.532</td>
<td>6.389</td>
<td>17.731</td>
<td>4.660</td>
<td>14</td>
</tr>
</tbody>
</table>

Figure 12.17: SPSS level two T Test

Again we can see that the P-Value is less than 0.01, and it is lower than our alpha value of 0.05. The same conclusion as for level one can be determined here also, that the mean difference between players choosing the shader areas over the non shader areas is statistically different and it reinforces the fact that we can completely reject the null hypothesis.

12.12 Conclusion and Analysis

From the above results, we can be sure that the proposed hypothesis is correct by the significance value. The vast majority of maps that show the players positions travelled do tend to show that players stayed close to shader hot spots. This was further reinforced by the standard error clearly proving the samples examined are representative of the entire population. From the above analysis, we can reject the null hypothesis and accept that players do tend to stay in shader locations much
longer than non shader locations.
Chapter 13
Benefits of Using Unity

13.1 Abstract

This chapter highlights the advantages of using Unity in game production and some of its hidden features that allow for easier game production.

13.2 Unity Remote

The set up shown in the deployment diagram in the Design of Game chapter was using a special application installed on the iPhone 4 called Unity Remote 3. This app transmitted player movements to the simulator. The screenshots in Figure 13.1 and Figure 13.2 shows this app.

Figure 13.1: Unity Remote App
This controller is great for evaluation purposes as I have demonstrated but it also facilitates during game production. When there is a new feature added to the game or something needs to be tested, then starting the simulator and the remote app can be a quick method of verifying. Usually, the game would have to be compiled and installed on the device, but by using the remote app it can be run directly.

13.3 Easy Interface

The Unity interface is great as menus are located where they should be and the menu structure is consistent with all other major modeling packages as shown in Figure 13.3.

The hierarchy tab at the bottom left is where the games objects are grouped according to their relations to one another. The project tab in the center is where all your materials and models can be accessed from. The inspector tab on the right is for accessing the currently selected game objects properties. The great aspect of these
Figure 13.3: Unity Interface
windows are that they are clear, simple and structured. If you wanted to change the color of an object, you would know to access the inspector tab and change the color from within there.

13.4 Choice of Languages

As mentioned before, Unity allows the developer to choose from using JavaScript or C Sharp. This is a great feature as it allows the developer to play to their strengths. I believe providing the choice to developers in selecting what language to use only increases the Unitys friendliness value to any new developer.

13.5 Built in libraries

As it was discussed in the Selecting the correct technology chapter, certain packages come with physics libraries and some dont. This was not the case for Unity as it contained built in support for physics and collision detection. This is extremely beneficial as one is not troubled in finding and linking third party libraries. The support is so well linked in with Unity, that as you script the required functions to override are available at your finger tips. There is no need for unnecessary import statements.

13.6 Global Variables

One of the strongest and attractive aspects of Unity is that variables can be made global and accessed from anywhere from within the projects file group. Usually, a global variable is accessible from within one script, but Unity allows the global variable from one script to be accessible to all other independent scripts. This is great if your want to loosely set a property without defining class dependencies.
13.7 Ready to use Functionality

Unity has many ready to use setups that are prebuilt. This includes setups for first person camera setup or third person camera setup. All that's required is to drag and drop the object into your project and start linking to it. The joystick setup was also prebuilt. These prebuilt setups enable a developer to jump start developing game prototypes and quickly move on with developing levels instead of setting up cameras and controllers.

13.8 Easy to Create multiple levels

The Unity engine has simplified the process of creating multiple game levels. Each level is treated as a new scene that can be created at any time and the character setups that need to be replicated to the newer level are easily transferrable. This simplicity of adding newer levels enables for quicker game production.

13.9 Conclusion and Analysis

Unity has reduced the complexity of game programming and I believe this is a fair statements. Since the industry of game production began, programming games in C++ was the norm. It has always been programming intensive where team developers have clearly set roles and are not allowed to maneuver between their skill sets. Unity allows for a team of developers or a developer to access every process of the game production and be part of it.
Chapter 14
Tackling Problems

14.1 Abstract

This chapter describes the problems that I encountered through the course of this project. I reveal my proposed solutions to these problems and the results they gave me.

14.2 Problem Solving

The project required data to be written to text files to be processed in Matlab. My first attempt was creating a stream writer object and linking it to a text file. As the player travelled in the game, the x and y values were written to the text file in parallel while the game ran. A problem was noticed when the values written to the text file were incomplete; the last few seconds of player traveling data were excluded from the write operation. This was due to Unity loading the next level and perhaps the write operation was on a lower priority thread. To solve this problem on getting a full text file of player visited locations, I had to change my algorithm on how values were written. Instead I added a couple of buffers for level one and level two where the values would be stored in real time. Once when the player completed the last level, a blank level was displayed with the title Game Complete. At this point, the values were written to the text files. On inspection, the list of values were complete and hence I solved my problem.
14.3 Writing Custom Scripts

Continuing from above, as I evaluated 15 players, there was a lot of data to be processed in Matlab. For the first couple of attempts I realized reading in all data and manually producing the maps would be tedious. To solve this problem, I wrote two Matlab scripts that would process data for the two levels. These scripts would read the data, produce the required map and add in locations where shaders are located. This solved my solution and a tedious task became very easy (scripts available on disk).

14.4 Tutorials

The beginning of this project, I did not have any knowledge on creating shaders. This was a massive problem. I solved this issue by initially reading up chapters in the books listed in the bibliography. As these sources are reviewed and contain valid information, I quickly grabbed key terms and processes that I would require for creating shaders, for example, terminology on normal, diffuse and height maps. After developing a critical understanding of what was required and what can be neglected for creating shaders, I started to research how to create the maps in Photoshop. By searching up for create normal map or create height map on YouTube, there were plenty of tutorials showing the procedures involved in creating these maps in Adobe Photoshop.

14.5 Access and Availability of Sources

The area I am researching into has a strong literature on Playstation and PC games where the power of the machines can handle shaders easily. The literature goes absent as soon as you search for mobile device performances on rendering high quality
graphics, there is not much. I believe this is due to mobile devices being still relatively new technology but have not previously had the processing capabilities to handle high amounts of data. I solved this problem by accessing articles that were relevant to mobile processing. This included research topics like image processing on mobile devices, etc. Clearly it is nothing to do with shaders, but still does demonstrate the mobile devices processing power.

14.6 Distraction into other areas

Unity is a massive games engine. There are many in built features that are tempting to start playing around with. One of the unexplored areas in this project was sounds in games and if they would be used in this game. Unity provides an excellent in built sound manipulation panel where the pitch, ambience and other sound properties can be changed. It was tempting to add sound effects into the game. When a player collected the ammo boxes, a sound could be played to confirm their collection. I was clearly venturing and researching into an area that would waste my time and it would cause me to deviate from the project timeline. Thus it was extremely important for me that I only focussed on the areas in Unity that were specifically for shaders. I solved this problem by staying on the projects timelines and not looking into other areas for unnecessary distraction.

14.7 Conclusion and Analysis

From the above, we can notice that there were problems in the production of this game. The initial way of tackling the problem is not the best approach as you have not thought of the best way of doing the job. Thinking and planning for these sorts of problems are difficult and require careful thought. I think taking a step back
and thinking about the amount of data that needs to be processed and proposing a solution to match the size of the task is best. This way you can cut down wasting time on repeating tasks and you can produce a piece of code where it can be reused for your own intents and purposes.
Chapter 15  
Future Work  

15.1 Abstract  
This chapter discusses other opportunities in continuing developing this project and future suggestions on how to improve this project.  

15.2 Complete the current Game  
As it has been mentioned, the networking requirements were discarded due to timescales. The game is half working with networking and multiplayer functionality. This can be seen in Figure 15.1.  

![Figure 15.1: Half Complete Controls](image)

As it can be seen, the buttons to start server and connect are all to do with
multiplayer setups and are half complete. The join match button is 60% complete but requires the correct transfer of controls to the second player. This transfer of controls was the problem I faced otherwise this functionality was complete.

15.3 iPad 2

When I finished planning for this project, the iPad 2 was released. At the time of release, I had already selected to use the iPhone 4 for my project. The literature report was created based on the iPhone 4. Designing a game for a smaller screen involves more careful thought in what's needed to be displayed in one screen view. The iPad 2 has a bigger screen, better resolution and more processing power. So for programming a game for this device requires a new set of objectives that encompass the extra processing power. It was regretful that I could not incorporate iPad 2 into my project but the option is there for future use.

15.4 C Sharp Advantages

Using C Sharp for programming allows the developer to gain access to .NET web services. This opens up the option for our remote game (client) to communicate with a server to retrieve player leader boards or game updates like textures or extra game characters. If the game was deployed to a wider market, a server and client setup would be crucial. Clearly a communication protocol like JSON or XML to pass data back forth from the server to the game would need to be implemented. The data that could be collected from a client and server setup would include a wider sample of the public. Hence, we would expect the sampled population to be even more representative.
15.5 The App Store

Continuing on the topic of wider release to the public, releasing the game on the App Store would improve the evaluation. This approach would only work only if the server and client side scripts were implemented. If there was no server or client side scripting, then releasing the game onto the App Store would just be for self pride. However, the App Store does provide the facility for customers to provide feedback on Apps. So setting up a framework for gathering responses from the App Store and evaluating them can add an extra dimension for evaluation.

15.6 Artistic Touch

Implementing shaders using image maps and fusing them into one material that is applied to an object is one way of producing shaders. The other way is to program shaders. This is extremely difficult and time consuming but a very elegant and powerful discipline to have. An artist would create a shader to enhance a scene instead of producing a more bumpier or 3D looking texture. A suggestion for a future project can be applying shaders to add artistic expression to a scene. This could be to set the mood, tone, etc. Finding a means of measuring the effects of this on players would also be intriguing.

15.7 Programming Shaders

If one does approach to program shaders rather than fuse image maps to create shaders, a key measure to make is if a programmed shader is more computationally efficient? If it is computationally efficient, it would be more appropriate as a future improvement to implement shaders via programming as it cuts out loading image bitmaps into memory and instead it directly creates the shader on the graphics
processor unit ready for use. This is a an interesting idea and is very valuable if measurements are made.

15.8 Conclusion and Analysis

For future work there is clear and practical suggestions from the above that can be set as captivating projects. With reflection, it is good at this point to draw a line on what I have done and why. The literature review laid out a clear Gantt Chart with project timescales and the software was developed according to these timelines. However, if one were to pick one of these topics from above and set upon developing them, it would advance on what i have researched upon. This project has been done according to what technology was most advanced at that point in time and how convenient it was for myself to complete the job at hand according to timelines. If I started the project with thought of continuing the project in the future, I would of programmed in C Sharp for accessing the web services, I would of programmed artistic shaders on a iPad 2 and would have accomplished the above set of suggestions. However, practically what is achievable depends on the complexity of the requirements and the set timescales.
Chapter 16

Conclusion

16.1 Abstract

This chapter is the conclusive chapter of the project. It summaries the main concepts discussed through the course of this thesis. A critical analysis if the correct decisions were made on use of technology in bringing the game to its current state. The significant findings from the evaluation session are reinforced by what was proposed in the introductory paragraph. This brings up the goals and aims that were set to drive the project and judgement if these objectives were met. Analysis on how well or badly this game achieves those objectives and if I have provided a platform for others to work from. Finally, the conclusion of this thesis and the answers it provides.

16.2 Correct Decisions

Initially when I was selecting what technology to use for producing the iPhone game, I was faced with what style of process (interactive or programing) on game creation I wanted and then, if required, picking the frameworks. I believe taking Unity as the approach to design the game was the best option. As a sole developer with no graphics or programming team, there was no possible way of me being the programmer, graphics designer and planner all at once. Unity abstracted most of these skills to a minimum and allowed me to get on with the task of creating the game and not micro managing sub tasks.
16.3 The Good and Bad

Through this masters project, I have went through good and bad. Good aspects of this project were getting the chance to use a fantastic game editor such as Unity. I believe this IDE and its presented style of game development will become the future. The fact that I was creating a game was in itself a great point. Critically, I believe any new comer into games development must be introduced and taught in Unity game engine before they are flung into the world of game programing. The bad point of this project was the networking aspect. As of now, there are images of the working game and clearly some networking buttons can be seen. This networking functionality is only half complete. The problem of producing a multiplayer game is firstly to instantiate the player then you have to transfer the camera and controls over to the newly created player object. The transferring of controls was the worst aspect of programming in Unity and getting it implemented was troublesome.

16.4 Uncanny Path?

One of the most intriguing and unsettling concepts that I researched upon was the Uncanny Valley [4]. This research is based upon how artificially created human robots and computer animations come to be disliked as they become more and more realistic. Relating this back to what this project is trying to achieve does raise certain issues. The paper does point out that its a particular problem with human replicas. Striving for photo realism from shaders is bad considering the hypothesis from the above paper. However, considering that its a problem relating to humanoid objects, then one can still apply shaders to other natural objects like trees, buildings, environment, etc. Then an evaluation procedure like the one carried out in the uncanny paper can be done in relation to this idea and measuring the human responses. It could reveal
some interesting results.

16.5 Findings

The statistical analysis of times that were collected from the evaluation session clearly reveal that players are spending more time in the shader areas than non shader areas. This is one way of analyzing the data. The player tracking data that was collected was presented as cluster maps that revealed the way the player travelled in game. To perform a fuller analysis of this data would be to check visiting patterns to the shader areas. However, the results from the statistical analysis clearly prove the hypothesis although the above analysis was not carried out.

16.6 Objectives Met

This game had a very simple in game objective where the player collected ammo boxes. This is such a simple task and not much thinking procedure needs to go into this. Hence, the user has enough mental bandwidth left for them to examine the surrounding environment. The shaders implemented in this game were displayed to grab attention and act as a player magnet. This was validated from the evaluations statistical results. However, a criticism of the final produced shader was its heaviness and was overwhelming. An artist would correct this by getting their artistic expression through balance and elegancy. As I am not an artist, I believe I have lost out on using shaders as artistic expression.

16.7 Ground Platform Established

I have proved it is feasible to create a game of some sort as a sole developer. If there were a team of developers, the possibilities are limitless. The way my game
is functioning as of now is the starting platform for any future developer who wants to take this principle of shaders influencing people to the next step. That could be implementing shaders in an artistic way. Another important aspect is that the evaluation process in this project has been hugely successful. This defines and reinforces the steps I had taken were the correct ones.

16.8 Impact on Field

As of now, the impact of this project onto a big games industry is small as the projects overall analysis still has the potential to be even more thorough and statistically more significant. The shader design will need to be reworked so that its presented more elegantly. The design of the level can also be designed to function closely with modern games. The number of participants have to be increased and data collected on them needs to include psychological analysis. Only after these improvements the results of this project can be taken seriously. However, as of now, I am the first to propose there is such a connection and identify suitable methods to evaluate such an idea. I have clearly demonstrated through the course of this thesis that the hypothesis is true and it leaves the door open to more advanced methods of analysis.

16.9 Thesis Conclusion

The backbone of this thesis is built upon the iPhone 4 and how it would be able to display high fidelity of graphics without any decrease in frame rates. During the project iterations it was quickly noticed that the iPhone 4 could only handle very limited amount of shader objects due to decreasing frame rates, consequently I was forced to use the Unity simulator. The introduction of this project clearly states the iPhone 4s performance on this matter is also being measured and it did perform
poorly. However with increasing processing power, the ability of smart phones to display high quality graphics will only get better and better.

This thesis has surveyed the different types of shaders that are available and concluded that in the current generation of devices, shader performance is very limited.
Chapter 17
Appendix 1
Bibliography


