I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

___________________________________________  __________
Name                                               Date
Acknowledgements

I would like to thank first and foremost my project supervisor Carol O Sullivan for accepting me for my final year project. I appreciate the support and continued assistance that she provided throughout the project duration. I would also like to thank members of the GV2 in Trinity College, Dublin for assisting me in my work, in particular Ludovic for his help with animations and Rowan in helping me with textures.

A big thanks also to my friends and family who each helped in their own way.
## Table of Contents

Acknowledgements ................................................................................................. ii

Table of Contents ................................................................................................... iii

Table of Figures ......................................................................................................... vi

Chapter 1: Introduction ............................................................................................. 1
  1.1 Report Structure .............................................................................................. 1

Chapter 2: Aims, Motivation and Background .......................................................... 3
  2.1 Aim .................................................................................................................. 3
  2.2 Motivation ....................................................................................................... 4
    2.2.1 Creating a game ....................................................................................... 4
    2.2.2 Animation .............................................................................................. 4
    2.2.3 Open Community Software .................................................................. 4
  2.3 Background ..................................................................................................... 4
    2.3.1 Software Tools ....................................................................................... 4
    2.3.2 Approach ................................................................................................ 5

Chapter 3: Gameplay .................................................................................................. 6
  3.1 Overview .......................................................................................................... 6
  3.2 Opponent A.I. ................................................................................................ 8
    3.2.1 Planning .................................................................................................. 8
    3.2.2 Implementation ...................................................................................... 8
    3.2.3 Included Features .................................................................................. 8
  3.3 Collision detection .......................................................................................... 9
  3.4 Shot Variance .................................................................................................. 10
  3.5 Game Structure ............................................................................................... 11
    3.5.1 Cut Scenes .............................................................................................. 12
  3.6 Game controls ................................................................................................ 12
    3.6.1 On-screen pushbuttons ......................................................................... 13
  3.6 Scoreboard Panel ............................................................................................ 13
  3.7 Camera angles ................................................................................................. 13

Chapter 4: Models .................................................................................................... 14
  4.1 Overview .......................................................................................................... 14
  4.2 Design Choices ............................................................................................... 15
  4.3 Model Pipeline ................................................................................................. 16
    4.3.1 Model Creation ...................................................................................... 16
# Table of Figures

<table>
<thead>
<tr>
<th>Figure 2-1: A Tennis Simulation Game</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3-1: Gameplay Bricks</td>
<td>6</td>
</tr>
<tr>
<td>Figure 3-2: Basic gameplay analogy</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3-3: Metabricks</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3-4: Court surface and net collision detection</td>
<td>9</td>
</tr>
<tr>
<td>Figure 3-5: Different shot characteristics</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3-6: Game Beginning</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3-7: Serving</td>
<td>11</td>
</tr>
<tr>
<td>Figure 3-8: Game Conclusion</td>
<td>12</td>
</tr>
<tr>
<td>Figure 4-1: Model Pipeline</td>
<td>16</td>
</tr>
<tr>
<td>Figure 4-2: Tennis Court Model</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4-3: Boneyard</td>
<td>17</td>
</tr>
<tr>
<td>Figure 4-4: AssimpView, incorrectly loading textures</td>
<td>18</td>
</tr>
<tr>
<td>Figure 4-5: Editing texture models using Paint.Net</td>
<td>19</td>
</tr>
<tr>
<td>Figure 4-6: Skybox</td>
<td>20</td>
</tr>
<tr>
<td>Figure 5-1: Principles of Animation</td>
<td>21</td>
</tr>
<tr>
<td>Figure 5-2: Structure of BVH animation files</td>
<td>22</td>
</tr>
<tr>
<td>Figure 5-3: Relating joints in a skeletal structure to a character model</td>
<td>23</td>
</tr>
<tr>
<td>Figure 5-4: Early Animation Model</td>
<td>23</td>
</tr>
<tr>
<td>Figure 5-5: Motion Capture using markers</td>
<td>24</td>
</tr>
<tr>
<td>Figure 5-6: Accessing the global root position</td>
<td>25</td>
</tr>
<tr>
<td>Figure 5-7: Accessing the rotations and translations</td>
<td>25</td>
</tr>
<tr>
<td>Figure 5-8: Troubleshooting animations</td>
<td>26</td>
</tr>
<tr>
<td>Figure 5-9: Walking, serving and flat shot animations</td>
<td>27</td>
</tr>
<tr>
<td>Figure 6-1: Tennis ball shadow</td>
<td>31</td>
</tr>
<tr>
<td>Figure 7-1: Man model</td>
<td>33</td>
</tr>
<tr>
<td>Figure 8-1: Mismatching pivot points</td>
<td>37</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

The computer games industry has experienced overwhelming growth over the past few years and shows no signs of slowing down [1]. Coupled with the meteoric rise of computing and processor capabilities the games arena has exploded. Increasingly large amounts of money are being spent on improving core performance features. Gaming production budgets are now approaching and surpassing those of major Hollywood movies, some stretching over the $100 million mark [2].

In attempting to create a well rounded game, this project seeks to examine some of the core considerations that go into production within the industry. It examines the feasibility of producing a game on a small scale and looks at some of the performance areas that would take a hit as a result of this. It also examines the open source community and its usefulness as a resource in facilitating the programmer.

There are many different aspects to consider when writing a game. Designers have a chance to show off their many artistic and creative talents in their bid to appeal to the player. It is in their interests to understand the key components that go into making a good game. In this manner they can get an idea of the tradeoffs that exist between the different criteria in a bid to appropriately focus their resources. E.g. there is little point in creating a game with incredible animation and graphics when the gameplay comes up short.

1.1 Report Structure

Chapter 2 – The aims, motivation and background describe the reasons for choosing the project as well as the motivation for doing so. It also gives background information regarding the software tools and the extent to which they were used.

Chapter 3 – Details all the core structure and gameplay features of the game. Describes the planning and implementation of these features and some of the complicating issues that resulted.

Chapter 4 – This section details the pipeline used in constructing the models and the manner in which they were integrated into their respective in-game positions. A description is given of the modelling tools used and the manner in which they helped.

Chapter 5 – Details the animation methods considered, the manner in which the chosen solutions were implemented and the many alterations that needed to be made.

Chapter 6 – Behaviour, Realism and Immersion was about implementing the physical models and extra features that contributed to the accuracy of the games simulation. It was about drawing the player further into the simulation experience.

Chapter 7 – This chapter talked about the experience of coding using the open source community. The feasibility and advantages/disadvantages were explored in attempting to reach an outcome.
Chapter 8 – An evaluation of the game’s performance and of the pitfalls and short comings that were observed throughout. Also included is the feedback from a test group and its performance against an industry title. This helped to provide a more rounded opinion of its strengths and weaknesses.

Chapter 9 – A Conclusion of the aims as stated in Chapter 2 and ideas for future work.
Chapter 2: Aims, Motivation and Background

2.1 Aim
My initial aim was to develop a 3D, player controlled tennis simulator which could be incorporated as part of the Metropolis project. As a graphics based project, special emphasis would be placed on such areas as animation, modelling and realism. These criteria would then be evaluated against a similar industry type game and an attempt made to understand the dynamics behind video game production.

In examining the ways in which a large budget release surpasses a low budget counterpart it would be possible to gain some insight into the feasibility of a simple one game production. As a means of levelling the playing field the open source community could prove useful in this regard. The advantages of this resource to the small scale programmer would also be assessed.

In the later stages of the project development it was decided to not go ahead with the incorporation into the Trinity Metropolis project. Instead, a heavier focus was placed on the gameplay aspect. This encouraged creative thinking in a bid to constructively add to the overall simulation experience.

Figure 2-1: A Tennis Simulation Game
2.2 Motivation

In searching for a possible project opportunity a number of environmental aspects within the Trinity Metropolis Project were considered. Many areas had been marked for improvement and there were numerous possibilities for further study. From the outset there was one project that proved to be of particular interest. This related to the tennis court areas located in Botany Bay.

2.2.1 Creating a game

Using programming as a tool to create and manipulate different scenarios and environments contains a certain lure and fascination to a relatively wide demographic of society. It is a common enough aspiration, especially for people of the younger “Nintendo generation” to try their hand at this profession. It would be of interest to study some of the areas of interest and challenges that are encountered in this line of work.

It was also of interest to study some of the overall processes that go into the creation of a game within the gaming industry and which aspects are of greater importance to focus on.

2.2.2 Animation

Animation has become a big production business in recent years as animators have had access to larger “technology budgets”. Companies such as Pixar are a testament to the value of good animation and have a lot to offer in terms of techniques. With the help of the Graphics department in incorporating such technologies as motion capture, this would prove to be an important opportunity to explore some of the ideas and processes that are considered for animation.

2.2.3 Open Community Software

There is a general understanding among programmers of the difficulties and caveats involved with using open source programming. Of particular difficulty is the ability to make externally sourced code compatible with the programmers own application. One motivation for this project was to determine the usefulness of the open source community as a viable means of contributing to an application of this spec.

2.3 Background

2.3.1 Software Tools

During the projects design phase there were a number of implementation approaches considered. Previous experience proved to be a useful tool in providing a basis for a number of different parts of the simulation.

The project was to be coded in C++ using the OpenGL libraries. This was an open community graphics library to which a certain degree of familiarity had already been established. One of the early initial challenges posed was the transfer from OpenGL to the “Ogre” graphics engine which was being used for the Metropolis Project. This transfer, however, never went ahead.

Google Sketchup used in combination with 3DS Studio Max were the modelling tools of choice. Paint.Net and AssimpView were used to help with textures and Microsoft Visual Studio was the coding platform of choice. Open source coding would prove to be a relatively alien experience.
2.3.2 Approach

Gameplay focus was the initial starting point of the project. This was deemed to be the most transferable aspect of the project as the algorithms could remain largely consistent regardless of the models or graphics engine that were to be used. Any progress in this manner would likely remain concrete and unmodified for the duration of the project timeline.

In the latter stages of the project it would be necessary to give more attention to the modelling and animation aspects of the game. Modelling demands would be largely subject to the type of model loaders which could be incorporated as would the physical laws be subject to change depending on the model dimensions. A certain degree of familiarly and certainty about the project outcome would be vital before undertaking such tasks as these.
Chapter 3: Gameplay

3.1 Overview

"Gameplay is the association of "Game rules", stating a goal to reach, with "Play rules", defining means and constraints to reach this goal [3]."

A game is typically most strongly defined by its gameplay. It can be classified as an activity defined by two elements: the rules and the outcome [3]. The rules of the game dictate the variety of possible actions that can be undertaken in order to achieve the result.

For the purposes of getting a clearer and more coherent view on what gameplay means within the industry and how it is approached, some research was conducted.

Gameplay is empirically seen as a central element within a videogame, and seems closely related to the game quality in the mind of many players [4]. There are generally two types of gameplay styles that a computer game can incorporate [5]:

- Game-based - The game is designed with a goal in mind. The player can win or lose and is generally marked on score.
- Play-based – There are no stated goals to achieve but the player is free to create and pursue his own.

The gameplay of a given title can be defined as the game’s core rules [5]. These rules can be represented as “gameplay bricks” which are present in all game types. The orange bricks represent the game rules and the blue ones represent the means in which these goals can be achieved. We can see how the blue bricks are influenced by gamer input while the orange ones are more associated with the end goal. The basic series of 10 bricks as illustrated below account for the majority of gameplay options within a given title.

![Figure 3-1: Gameplay Bricks](image)

The game rules that are defined by these bricks can be broken down into a condition and an action, i.e. if something proves to be true... then carry out the appropriate action. This sequence of events can be represented by a series of “if” statements and corresponding actions. This was also representative of my approach to gameplay and in some of the features I chose to add.
Figure 3.2 illustrates how these game play brick go into constituting a series of familiar gameplay scenarios.

When a game is in the design phase it is very common to see repeated pairs of bricks emerge in repeated association with each other. They give rise to some of the major genres that we see today and greatly influence how the games are classified. Examples of such “metabricks” are illustrated below in figure 3.3.

Introducing and meditating on gameplay features proved to be an opportunity to interject some creativity into my game. Even based around the restriction of creating a tennis based experience there were a number of liberties and options that it was possible to expand upon.
3.2 Opponent A.I.
A.I. can often be the most challenging design feature that goes into game production. There is always limitless scope for improvement and as software developers are limited in terms of resources such as time and computing power there is never such a thing as a perfect A.I. Decisions would have to be made upon the best and most appropriate features to include within the time frame.

The creation of a player A.I. proved to be the most enjoyable and entertaining experience. While working within the scope of tennis rules the following A.I. capabilities were considered: Ability to return a shot, choice of aiming, player shot selection, player speed and altering difficulty levels.

3.2.1 Planning
In the initial stages of the game it was important to plan out all the A.I. capabilities that were to be included. The initial stages required both the greatest amount of foresight and creativity as it was necessary to be realistic about what was possible to achieve. One helpful method proved to be the conceptualisation of the desirable characteristics which would most add to the challenge and element of enjoyment. It would be beneficial to incorporate a varying degree of gameplay situations in which the opponent could react to. This would lead to a lesser amount of monotony in the style of play.

3.2.2 Implementation
The next stage of the process was the implementation of all these features with relation to code and program layout. This would prove to be an ongoing and continuously restructuring process in which program flow was the most important factor. A host of different descriptive members and variables were included which were required to be updated at precise moments in the code. IF and CASE statements were the principal means of implementing the “intelligence”. This proved to be the most straightforward and intuitive way in which to influence the opponent character. The positioning and trajectory of the tennis ball would be monitored in order to influence its due course of action. Other features such as player skill and location of player controlled character would also influence the decision making of the NPC (non-player character).

One way to facilitate coding process was to create a number of different Boolean variables which were used to describe the motion and location of the ball at a given time. For example, the variable “almost_end_of_player2_court” when true, was quite clear in signifying that the ball was close to the end of the court on the NPC’s side. This variable was toggled to be true and false based on a monitoring function “tennisball_intelligence()”. In this example I could set the opposing player to attempt to swing the racket to hit the ball once the trigger had been set. This coding style proved to be a much easier and intuitive solution to the constant re-evaluation object parameters each time an action was to be performed.

3.2.3 Included Features
The following AI features were decided upon:

**Altering difficulty levels** - Based on a selection that the player makes at the beginning of the game, the player can choose to play against an easy, medium or hard skilled opponent. Differences in opponent skill reflect differences in opponent: accuracy, aiming direction, number of shots available in arsenal and speed.
Selective aiming – The NPC monitors which side of the court the player’s character is on and chooses its aiming direction based on this. Continuously computed random variables go into creating a weighted average system which is used to decide upon the outcome of these A.I decisions. In the case of a harder skill level there was a higher probability that the ball would be aimed at the opposite side of the court.

Changing shots – In the hard skill level I allowed for the ability of the NPC to change the shots within his arsenal. Again, the shots were decided on a weighted random basis. This would add to the challenge for the human controller as he was thrown a varying array of slice, top spin, power and flat shots.

Opponent movement – Most importantly the opponent would have a varying speed in which it could be possible to move about the court. The speed was dictated by the NPC’s skill level in so far as the harder skill levels had greater speeds. A variance of player animations was also decided upon based on the incoming trajectory of the tennis ball. In this way the NPC would carry out different kinds of forehand and backhand animations to mix up the style to a certain degree.

3.3 Collision detection

Within the context of a tennis court, collision detection proved to be a straight forward idea to implement. This was done through the creation of a monitor function “collision_detection()” which kept track of the ball in every frame to detect when the ball was out. Collisions with the net, player baselines and either side of the tram lines were determined to be a foul. In each case, the point was appropriately conceded and the game returned to service state. It proved especially important to use Boolean descriptors in order to keep track of: general location of interest, last person to hit the ball, serving turns, etc... Sound triggers were added to indicate the collision moments.

![Figure 3-4: Court surface and net collision detection](image)
3.4 Shot Variance

In order to play the game at a competitive level there are a certain number of shot types that the tennis player must master [6]. Some of these include: standard forehand/backhand, top spin, slice, serve, etc... Each shot within the arsenal has an appropriate moment in which it can be played.

This was an important gameplay feature present within tennis that was essential to build upon. Five shot types were created: flat-shot, top-spin, slice, power and serve. Each shot was described through a number of different physical characteristics whose parameters were specified by means of a 2D array. Depending on the type of shot selected, there was a variance in power, accuracy and angle observed. The characteristics of each shot were appropriately altered to mimic real world characteristics and values.

```
float shot_type[5][7] = {{1.2, 1, 1, 1, .25, .4, 0, }, //flatshot
{0.9, 1.5, .8, 1, .5, .5, 0, }, //topspin
{0.8, 1, 1, .8, .1, .8, 0, }, //slice
{2, 1, .7, .7, .15, .25, 0, }, //power
{1.5, 1.5, 1, 1, .1, .5, 0, }, //serve
```

Figure 3-5: Different shot characteristics

The flat shot was the default and provided the player with a guaranteed means of getting the ball across to the other side within the court boundaries. It was the safe, reliable choice.

The top-spin shot provided the player with greater angular capabilities which could prove useful in hitting the ball out of the opponents reach. However it also had decreased accuracy capabilities.

The slice is a shot in the tennis player’s arsenal that is used to slow down the pace of play. This is a defensive shot used to give the player more breathing space if he is being run from one side of the court to the other. In this respect, the power of the slice was toned down. It had good accuracy and poor angle.

The power shot is used at the player’s own risk. It epitomises a common gameplay component which is weighing up risk against reward. It greatly accelerates the speed of the tennis players shot and thusly increases the chance of scoring a shot. This however, is usually offset by the greatly reduced accuracy in the x and y directions.

The serve was the only means of beginning a player’s service and was of moderately increased power with a small accuracy deficiency.

Each shot was constructed with a specific gameplay advantage in mind and each weighed a certain amount of risk against chance of success.
3.5 Game Structure
The game layout had a classic beginning, middle and end structure to it. In the beginning the player was prompted to pick a skill level. The middle part consisted of the game itself, dotted with numerous serves and the end was marked by another scene.

The game was structured in a way which is familiar to all tennis players. The usual rules of tennis applied. The serves altered every other game. The classic tennis scoring system of games, sets and matches was implemented with the optional number of sets to win.
3.5.1 Cut Scenes
Three cut scenes were included as part of the game structure. They limited available actions of the player to one or two specific triggers and provided a break in the run of play.

The first related to the game beginning as the player is prompted to choose a skill level. Upon choosing, the game progresses to the serve scene. To implement this it was necessary to set up an initial viewport covering the entire application window and text was displayed using the “GlFont[]” library.

The serve scene resets the players to their respective places on the baselines. Play is begun once the player serves the ball. Serving alternates between every consecutive game so each player has equal share in serving turns.

The final cut scene marks the end of the game. Upon reaching the final set score the game is declared to be over. The winner is noted on the score panel and both players walk towards each other to the centre net to congratulate each other on yet another fantastic game of tennis.

3.6 Game controls
With the aim of providing a greater means of control and supplementing a wider amount of gameplay options input from both mouse and keyboard was introduced. The directions used the classic “WASD” keyboard combination and the push buttons on the side panel were used for shot selection. This is an intuitive setup for people who had had practice at gaming.

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>Move forwards</td>
</tr>
<tr>
<td>A</td>
<td>Move left</td>
</tr>
<tr>
<td>S</td>
<td>Move backwards</td>
</tr>
<tr>
<td>D</td>
<td>Move right</td>
</tr>
</tbody>
</table>
E - Hit the ball
J - Aim left
K - Aim centre
L - Aim right
Number pad - Camera controls
Mouse – Change shot

Mouse usage was a key component in introducing a greater amount of functionality. The keyboard was the standard, base method for input up until the point where there were an ever increasing amount of options made available to the player, leading to the demand for an easier means of control.

3.6.1 On-screen pushbuttons
It is generally accepted that the addition of mouse control in a typical computer game can vastly add to the degree of control that the player has [7]. A panel of push buttons was added to give the player a choice in the return shot type that s/he desired.

Initial attempts to implement the buttons in the main viewport were un-successful. The “GlPerspective()” function introduced a mismatch between a given location of a button with respect to the x, y pixel coordinates and its location in the 3d environment. The buttons would also clip through the tennis court or surrounding models as the camera viewpoint was changed. For this reason it was necessary to implement them using a separate 2D orthographic viewport.

3.6 Scoreboard Panel
The scoreboard side panel was introduced partly for greater player immersion and partly for presentation purposes. The panel was set up using a different viewport on the right hand side of the screen on which an orthographic view was displayed. This was necessary in order to view the panel components in the correct, top down manner.

The side panel displayed other important information such as: NPC’s skill, current point score, game score, set score and the winner (at the end of the game). The text was displayed with some difficulty using updating stringstream objects and the GlFont libraries.

3.7 Camera angles
Camera angles provided a variance of further gameplay opportunities for the player. The camera could be manipulated freely around the scene using the number keys or set to a series of three set camera views, ideally placed for different key perspectives.

The first camera angle provided the player with an overall view of the tennis court and was the preferred for normal play. Another gave an up close and personal view of the action with the intention of mimicking the viewing position of the player’s character. This added to the immersion factor.
Chapter 4: Models

4.1 Overview
A large component of computer game design is focused on the creation of 3D models. They are the essential building blocks on which a 3D game is founded. Every object and environment within a game needs to be illustrated through the means of a 3D model so game studios dedicate a lot of resources into getting them right. It is essentially a mathematical representation of a real world object and can be manipulated and modified through the use of various software packages.

The process of adding an object as part of a game generally involves three stages [8]: "modelling", "viewing" and "rendering". The modelling process is primarily concerned with the geometry of the object. Viewing is the task of introducing the laws of perspective to help create the illusion of a 3D space from a 2D plane. The rendering determines how the model is actually displayed within the game itself.

In the gaming and the film industry there are two primary types of 3D models that are used: [9]

- NURBS surface – This is a smooth surface created through the use of Bezier curved and boasts the highest amount of mathematical precision.
- Polygonal Model – Polygonal surfaces are constructed out of simple shapes called object primitives and are the most prevalent in use in the gaming industry.

Textures and shaders are also to be considered. Shaders dictate how the object is to be displayed and controls the way in which it interacts with a light source. Textures are 2 dimensional image files that are mapped onto the model’s 3D surface.

In the gaming industry it can be necessary to include a lot of descriptive parameters when talking about models. There are also a whole host of principals to be considered when adding an object to a game. These include: Realism, Detail, Functionality, Scale and Proportion, Appeal, Volume, Exaggeration and Weight [10]. When properly considered, these help in integrating the models to a better fit within the game.
4.2 Design Choices

Before beginning, an understanding of the various in-game models which would be employed was required. This would encompass all the basic features of a tennis court, tennis apparatus, background objects, features, player characters and more.

When it came to designing the models themselves there proved to be a lot of different software tools and packages to choose from. Google Sketchup proved to be the tool of choice based on previous familiarity and experience. While not necessarily the most feature inclusive package, it was simple and easy to use. It also had the added advantage of having a large user database of uploaded user content. This would prove to be a very important resource.

A necessary part of my pipeline would be the conversion of .skp files (Google Sketchup format) to the 3DS file format. In this regard, 3DS Studio Max and Babel3D online converter proved to be useful tools. The conversions using Babel3D proved to be a little cleaner but for the majority of objects 3DS studio max was used as small object manipulations were required.

Having played around with 3DS Studio Max, it proved tempting to incorporate a 3DS max model loader as part of my game. There were a multitude of heavily detailed 3ds max models online which would greatly improve the aesthetic value of the game. Attempts were made to achieve this but were later aborted due to the complexities involved in doing so. To this extent the existing 3DS model loader which was used in previous graphics labs would prove to be sufficient. There existed a multitude of 3DS files online and these would be more than adequate for the simple simulation requirements.
4.3 Model Pipeline
The programs and software packages used for the model pipeline are illustrated below in figure 4.1.

Figure 4-1: Model Pipeline

4.3.1 Model Creation
Google Sketchup was the model editor of choice. It had an easy to use interface and the fact that it was a familiar tool would prove to be a great bonus. It has the ability to create models to a high standard of accuracy and texture them as desired. This proved to be especially useful when it came to the construction of hierarchal structures. At this stage much time and effort was placed into the modelling process. This would later prove an unfruitful labour as the quality of the models was repeatedly compromised during the import and export processes. This might have been avoidable with a greater amount of foresight.
For the character structures it was necessary to create a series of “bones” to correlate with the separate joint structures which were described in the BVH files skeleton. To account for all the joints it was necessary to construct 17 different types of bone objects. The construction of these bones was carried out through the part by part dissection of an adult human male model that had been pulled off the Google Sketchup database.
4.3.1 Object Conversion
The Autodesk tool, 3ds Studio Max is generally acknowledged as a very powerful means of model creation and is of wide use within the gaming industry. Had artistic merit held greater importance it would have proved a lot more beneficial to achieve a greater degree of familiarity with it. It still proved to be a very useful tool in many ways.

One such way was the conversion of the models to the correct format. The tennis simulator made use of a 3DS loader and naturally the assets had to be converted to that format. 3DS Studio Max provided the ability to import files in the .skp file format and to export them in the .3ds format. There was the unfortunate consequence of the importing and exporting process changing some of the texturing and object parameters along the way. This could be fixed to some extent through altering the model parameters within the intermittent .3ds_max format.

The online resource 3DBabel also proved to be an important tool in this process. Some objects, as was the case with the tennis racket and the tennis ball, were not required to be altered before use in the game and to this regard Babel3D was used as it provided a better quality of model conversion.

4.3.2 AssimpView
AssimpView is a freeware model viewer which provides the programmer with a useful interface in which he can freely view models in a wide variety of formats. It proved to be a very useful tool in the debugging of some of the conversion errors which resulted from the many imports and exports. During importing and exporting from 3ds Studio max the object textures and looks tended to change dramatically. Further changes to the appearance were made upon viewing the models in game. This tool provided me with the ability to understand where some of the un-wanted changes were taking place and helped in narrowing the scope for finding a solution.

Figure 4-4: AssimpView, incorrectly loading textures
With respect to the tennis court model this proved to be especially useful. Texturing of the tennis court walls and net proved to be very problematic. There was no transparency within the model once it was being loaded up in game and there were a wide variety of issues that could have been to blame. The model viewer isolated the problem as the textures holding the alpha channel (transparency channel) in the wrong place. From here a program such as Paint.Net was used to alter the texture images in order to make the appropriate channel appear as the transparent one.

### 4.3.3 3DS Model Loader

The 3DS model loader [11] which had been used throughout the graphics lab was required for the reading in of the object files, storing of the information and having appropriate functions available for the rendering of the objects within the game.

There were some limitations imposed by the model viewer in terms of texturing. The only acceptable formats were .tga and .bmp and in this regard the images only loaded when their dimensions corresponded to a value that was a power of 2. For this reason it was necessary to use conversion and editing tools within my picture editor Paint.Net in order to configure them directly for use in the game.

![Editing texture models using Paint.Net](image)

### 4.3.4 Fixing the models in game

Translating, rotating and scaling proved to be a major process in the model pipeline. Depending on the source they came from the models could have vastly varying sizes. It was required to make them all proportional to each other above anything else.

The tennis court was the platform upon which every other model’s position and dimensions were to be made relational to. For the sake of simplicity it was aligned along the z-axis with the x-axis creating a rough bisection in the z-plane. The court parameters were recorded through taking observation measurements of small objects translated to various key areas within the scene. The tennis player’s locations were fixed according to these values as were the in-bound court constraints.
The ensuing translation and rotation of the bones within the human structure proved to be quite a painstaking process. Each bone had already been given a translation relative to its parent as described in the .bvh file but it was necessary to carry out further translations. This was due to the fact that the pivot point of the object did not correspond to the pivot point of the animation bone. Such a process as this would have been avoided should the pivot points have not been reset during the export process from 3DS Studio Max.

4.4 Skybox
A skybox (or sphere or cylinder etc..) is typically used within the games industry for background simulation [12]. The box object has textures applied to the inside faces and the object is then drawn around the player. The box textures will then remain in full view of the player no matter which way he turns, thus simulating the effect of a background.

The skybox and accompanying grass texture were important presentation features which really helped to introduce a more realistic and engaging experience for the player. Previous attempts with plain colours for the background sky and grass objects had resulted in a serious presentation hit. It became a major priority to implement a skybox. A textured background also helped the player to get a better locational understanding of the scene.

The skybox worked through loading in 6 different images to an image loader. For my purposes the images were required to be of a set size and of a set bit depth. Once the images were loaded in they were to be combined together to form a cube. A number of different options were made available in terms of image loaders, skybox constructor and image types. It was necessary to debug and ensure all components worked harmoniously before the end result could be observed.

![Figure 4-6: Skybox](image)
Chapter 5: Animation

5.1 Animation in gaming

Character animation has become a central element in the modern games entertainment industry [13]. In recent years, coupled with the continued increase in the hardware and software sophistication, characters are becoming increasingly complex. A higher level of detail can be achieved in both modelling and animation standards. However, due to the huge demand that is placed on the processors in real time there is still an upper limit to the amount of improvement that can be achieved [14].

One of the important considerations of a modern games designer is to make the characters within the game relatable to the gamer and in some cases for them to form an emotional attachment. This can be accomplished through the manner in which the character is animated. There is an increasing potential for generating this type of player-character empathy through the greater amount of character subtleties that are now being made available [13].

As opposed to movie animations, game animations are created in loops. This is due to the fact that the game has no set viewport or series of actions as it is based on unpredictable player input. This unpredictability of character actions is the main reason for differences in quality between animation in games and the cinema. Developers must create environments that provide players with a 360 degree field of vision at every point. The way animators deal with this unpredictability is to create animation loops. The idea is to make one loop flow into the next in a seamless integration.

Many animators consider the 12 principles of animation when working on a major game or movie title [15]. These can be summarised as: Squash and stretch, anticipation, staging, straight ahead action and pose to pose, follow through and overlapping action, slow in and slow out, arcs, secondary action, timing, exaggeration, solid drawing and appeal. Each of these adds to animation experience in its own way and should be taken into consideration when constructing any scene.

Figure 5-1: Principles of Animation
5.2 Planning

Animation proved to be the main focus of the project and took the greatest amount of time to implement. Previous attempts from earlier graphics projects had been time consuming, arithmetically challenging and had less than satisfactory results. It was decided that to strive for greater realism a different approach would be needed. A system was settled upon to make use of previously captured animation data structures which could be freely used within the game. In this respect my project coordinator Carol O Sullivan suggested the use of a BVH loader in order to load in some .bvh animation files. These files contained information about displacements and rotations of joints in a skeletal structure on a per frame basis (as illustrated in figure 5.2). These motions, when played out, would simulate a certain action or human behaviour.

![Figure 5-2: Structure of BVH animation files](image)

It would be necessary to model a series of bones around this skeletal structure with these key joints in mind. Each bone (with the exception of the root bone) would have a parent bone associated with it. The bone would translate and rotate with respect to the parent bone, greatly simplifying the overall process. Due complications and greatly increased workload which would result from creating and implementing a second animation structure it was of great interest to ensure the sourced animation files were of a consistent structure.
In order to help gain insight into the different movements depicted by each animation file a simple hierarchal structure made from cylinders, spheres and cubes was constructed. While not included as a final game feature it proved important in understanding and appropriately modifying the various depicted animations.
5.3 BVH File Loader
After initial attempts to configure a bvh file loader from a Oshita Labs [16] proved unsuccessful I was given an alternate by the Trinity Graphics Lab. The code consisted of a number of object classes. When tied together, these constructed the bones of a BVH file loader which would read in and store animation data. The end result was that upon program initialisation a skeleton object was created which possessed numerous data members. These members reflected the translation and rotational state of the bones with respect to their parents. The overall position of the object with respect to the environment was stored as part of the root bone data members.

The next issue was to load in and associate multiple animations as part of the tennis player’s hierarchal structure so that different movements could be performed. This was done through the creation of an array of skeletal objects. Upon the initialization of the project, files would be opened and fed in to the data members of these objects. Then, depending on the motions that were triggered during gameplay, the relevant skeleton and animation members would be accessed in order to describe the appropriate movement. It was decided to load all of the animations on program initialisation as it was a computationally expensive procedure and doing this in real time would have resulted in a vastly deteriorated frame rate.

5.4 Motion capture
When looking to include animations as part of the game, online resources were the first port of call in deciding what was available. There existed a huge number of files regarding general motion and motion in other particular areas but surprisingly little concerning tennis. Due to this unfortunate situation it was necessary to source the animations through alternate means. Motion capture was one way of doing this.

Motion capture is defined as "The creation of a 3D representation of a live performance" [17]. There are many different means of capturing this live performance data but in the case of the Trinity Graphics Department an optical system was used. Most optical systems are driven by tracking marker positions which are fixed to various parts of the human person [17]. This information is then used to describe the motion of a 3D skeleton structure which can be used in games.

![Figure 5-5: Motion Capture using markers](image-url)
With the kind assistance of the graphics department a number of tennis animations were provided which could be incorporated as part of the project. These included: walking, forehand and backhand motions (including variations) and serving. It was now possible to supply the tennis characters with all the basic motions which they needed in order to have a basic amount of functionality.

5.5 Playing the animations

For each animation sequence there were a number of variables which affected the positioning of the joints within the players: The global translation, the local rotations and translations of the bones and the frame number. All of the translations and rotations were to be considered in the x, y and z directions.

A function was needed to take care of the global position of the character. The global position coordinates for a given frame were accessed through the global position of the root bone in the skeletal structure. These were updated on a per frame basis in order to provide for a slow and steady movement in a given direction.

\[
\text{void importedModel:}: \text{updatePosition(float x, float y, float z, int frame_number, bool endanimation, int dir, int animation, int skill)\{}
\]

\[
\text{int scale}=600.0*(\text{skill}^{0.2})+1; \text{int scale}=500; \text{startx} = \text{skeletonArray[animation]}->\text{getGlobalPosition(frame_number)\.x}; \text{starty} = \text{skeletonArray[animation]}->\text{getGlobalPosition(frame_number)\.y}; \text{startz} = \text{skeletonArray[animation]}->\text{getGlobalPosition(frame_number)\.z}; \text{endx} = \text{skeletonArray[animation]}->\text{getGlobalPosition(frame_number)\.x}; \text{endy} = \text{skeletonArray[animation]}->\text{getGlobalPosition(frame_number)\.y}; \text{endz} = \text{skeletonArray[animation]}->\text{getGlobalPosition(frame_number)\.z}; \text{change in x} = \text{endx}-\text{startx}; \text{change in y} = \text{endy}-\text{starty}; \text{change in z} = \text{endz}-\text{startz};
\]

\text{Figure 5-6: Accessing the global root position}

The second function was constructed to take care of the local rotations and translations of the joints. This function took in a series of descriptive parameters in order to ensure the correct joint was moved the appropriate amount in a given frame.

\[
\text{void importedModel:}: \text{renderFigure}(\text{const std::string & joint_name, int frame_number, int scale, int animation})\{\}
\]

\[
\text{int t}=15; \text{// xaxs 50} \text{int t}=50; \text{// yaxs 50} \text{int rot3} = \text{frame_number}-0; \text{glTranslated}(\text{skeletonArray[animation]}->\text{findBone(joint_name)\.getOffset()\.x}, \text{skeletonArray[animation]}->\text{findBone(joint_name)\.getOffset()\.y}, \text{skeletonArray[animation]}->\text{findBone(joint_name)\.getOffset()\.z}); \text{glRotated}(\text{rotated(skeletonArray[animation]}->\text{findBone(joint_name)\.getOrientation(frame_number)\.x}, 1, 0, 0); \text{glRotated}(\text{rotated(skeletonArray[animation]}->\text{findBone(joint_name)\.getOrientation(frame_number)\.y}, 0, 1, 0); \text{glRotated}(\text{rotated(skeletonArray[animation]}->\text{findBone(joint_name)\.getOrientation(frame_number)\.z}, 0, 0, 1); \text{glScale}(\text{scale}, \text{scale}, \text{scale});)
\]

\text{Figure 5-7: Accessing the rotations and translations}

Within the game itself the animations were triggered through various different means. In the case of the player controlled character, more often than not, the animations were triggered by a key press. On the other hand the NPC animation sequences were triggered based on the state of certain variables such as ball and player proximity.
5.6 Fixing the bugs
Correcting the animations proved to be one of the most laborious tasks and there were many unexpected results observed.

5.6.1 Starting, Stopping and Looping
In the context of the game it was important to provide options for sudden changes in animation, speeding up the animations, looping the animations and finding appropriate times in which to start and stop them. These problems arose due to unpredictability in the means in which they could be triggered.

To solve the problem of looping, starting, stopping and changing animations I introduced a variable which set the animations to run continuously when set to positive. This variable was triggered at key moments such as a key press or triggered change in direction for the AI player. Through controlling this variable it was possible to keep the animations running in a smoother manner.

5.6.2 Unwanted Displacement
For some animated sequences there was an overall displacement observed between the first and last frame of a motion sequence in the y-plane. This had arisen from inaccuracies in the motion capture. A phenomenon developed where characters would sink through the ground after a series of repeated actions. This also was also reciprocated for movements along the z-plane causing the player to leave the baseline and wander closer or further away from the tennis net. For the case of a few trouble animations it became necessary to discard any overall movement in a given plane.

5.6.3 Orientation Problems
Each animation file had relative differences with respect to the directions they were played, the frame rate they were played at and the rotation of the player during this motion. Some of the motions would be played out relatively quickly while at the same time a simple forehand motion could take a full rally cycle to complete. For this reason it was decided to troubleshoot this through using a 2D array to hold all of the necessary alterations that were to be made in making them consistent with one another.

```c
float animation[11][3] = {
    {0, 2, 2},
    {180, 8, 2},
    {-90, 8, 2},
    {180, 8, 2},
    {-90, 8, 2},
    {-90, 4, 2},
    {-90, 4, 2},
    {-90, 8, 2},
    {180, 8, 2},
    {-90, 8, 2}};
```

Figure 5-8: Troubleshooting animations
5.7 Included animations

A series of 10 animations was used in the end which was included as part of the game:

**Walking** - This animation was directionally dependent. It was the most used animation sequence in the game and for that reason it was essential to iron out all the bugs.

**Serving** – There were two serving options available to the characters. These were played at the beginning of each point as part of the serving sequence.

**Forehand variations** – The player was triggered to do a forehand motion if the ball was approaching from the forehand side. The variations of this were the one step and two step forehands. These animations were triggered if the ball trajectory came in at an increased angle. The character would therefore be required to carry out either one or two side steps before hitting the ball. If an angle greater than these animations was observed the player would simply continue walking in the prevailing direction in an attempt to reach it before the ball crossed the baseline.

**Backhand variations** – The backhand motions were used when the tennis ball was approaching the opposing player from the back hand side. The variations were used in similar circumstances as above.

![Figure 5-9: Walking, serving and flat shot animations](image-url)
Chapter 6: Behaviour, Realism and Immersion

This section details how the physical models created within the game dictate the behaviour of the objects and contribute to an enhanced sense of realism experienced by the gamer.

6.1 Physics in computer games

It is difficult to give a concrete definition on what is meant by immersion but we can identify the barriers to immersion as being a lack of coherence within the gameplay [18]. The introduction of physical models and laws into a gameplay scenario can greatly add to the realism experienced by the player. Adequate graphics are also an important feature in this respect.

Some of the main principles which reoccur across most games titles and which are almost independent of genre include: gravity, elasticity, light and sound [19]. It is easy to understand that as these simple laws are observed on a day to day basis that they need to be replicated to a very high standard before the game is deemed to be realistic. For the most part, these laws can be stated as they are observed in the natural world as we expect to see things behave as they should. In some exceptional circumstances these laws can be altered in order to exaggerate an event or to perhaps mimic a change in environment.

As a programmer it is important to use intuition to incorporate the laws of physics where necessary in order to add to the overall experience.

6.2 Getting the measurements

To add to the experience and realism of the game it was intended to make the measurements and speeds as representative to real world values as possible. Correct measurements had a strong initial importance, as to begin with, it was the aim to incorporate the game as part of the Trinity Metropolis Project. It was therefore necessary to have the game dynamics made transferable to a separate scenario within the Metropolis environment. Differences in object dimensions could potentially give rise to problem scenarios.

6.2.1 Dimensions

Units of length, as measured by different software suites vary greatly in definition. There was little way of knowing how large or small a model would appear when introduced into the game for the first time. For this reason measurements were taken in-game. Small objects were translated and rotated about the tennis court in order to match the positioning of key tennis court parameters. Once the correct position was obtained the parameters were then defined and stored for later use. In this way dimensions for the: baselines, tramlines, surface depth, net height and net location were taken. The court model itself was constructed based off official statistics as was important for the principal model of the game.

6.2.2 Velocities

With respect to measuring players speed, a different approach was adopted. Timed runs were taken across the various key traversable distances within a tennis court. The idea was to get an accurate measurement for timing the players speed while at the same time using different sets of values to correlate with different opponent skills, i.e. a difficult opponent would be physically fit and be able to run faster. Runs were made from baseline to net, tram line to tram line and other various points.
Based on the fact that the game has a large part to do with acceleration and starting and stopping, it would be unrealistic to assume an average speed measurement to use this in time calculations.

Although initial aspirations encompassed a number of different speeds for the opposing tennis player, these were later scrapped. This was partly due to a shortage of captured animations for jogging and running. However the times were still used to a rough extent.

In terms of tennis ball speed, it was required to incorporate a number of different values. These correspond to the varying velocities of the shots within a player’s arsenal. For example, the slice shot is a slower pace than the flat shot which is in turn slower than the serve. The varying speeds were gauged from existing tennis games such as “Top Spin” (as little or no research exists in this area) in order to obtain accurate simulation speeds.

6.3 Physics simulation
It was my intention to incorporate some basic physic principals into the game with respect to the tennis ball. Ideally the ball would be modelled on all the physical principles and all the parameters would be set beforehand. The usual rules of momentum, gravitational pull and coefficient of restitution, friction, etc would apply. However, with respect to section 6.1 all scenarios observed proved to be under the umbrella of “exceptional circumstances” and modifications were needed to be made to these rules in order to improve the functionality. The focus shifted towards accurate simulation aesthetics as opposed to concrete physical law implication. Certain shortcuts had to be taken.

6.3.1 Gravity
Gravity was implemented as a simple force that acted upon the down velocity of the ball along the y-axis. An amount as determined by “GRAVITY” was deducted on a per frame basis from the y-velocity and this made the tennis ball dip to hit the court on the opposite side as it carried over the net.

6.3.2 Momentum
One of the most challenging problems was maintaining a realistic momentum model of the tennis ball over the course of a series of shots, especially when there were varying shot types involved.

The initial model envisaged a momentum in the x, y and z direction for the tennis ball. Upon striking the ball at either end, the ball velocities would change, but only relative to the previous direction from which it approached. In this way a fast approaching ball would instantly lose all its momentum and therefore would make for a more realistic flow of play.

However, this strategy had more than its fair share of problems and was soon scrapped. The ball trajectory would need to be carefully planned in order to keep it on a realistic path and this was proving to be a very difficult task. Because the tennis ball momentum operated on an incremental basis it could continuously speed up and slow down to the point where it approached unrealistic speeds. Controlling the tennis ball along the x axis also proved to be difficult as its trajectory would frequently balloon off course.

To this end, I found it necessary to reset some of the positioning and velocity values of the ball on approach to the baseline. This ensured a continuous run of play without unexpected or unrealistic
outcomes. I found that it didn’t become immediately noticeable to the eye and therefore added to the degree of player immersion over the previous scenario.

6.3.3 Elasticity
Within the context of my game, elasticity refers to the degree in which the tennis ball bounces back from the court surface. Part of the operation of my monitor functions were to indicate to the tennis ball as to when it had collided with the ground. When this event had occurred the ball velocity would rebound in the opposite direction to a degree specified by the “COEFFICIENT_OF_RESTITUTION” parameter. The co-efficient of restitution refers to a physical value which describes the loss of energy observed by an object on the plane of approach when it strikes another. It usually depends on the elasticity of the object as bouncy objects tend to rebound with a greater force. This was successfully implemented and caused shots such as top spin to rebound to a greater height than that of slice or flat shot.

6.3.4 Sounds
Initially upon looking to incorporate sounds as part of the project OpenAL was the software package of choice. This has dedicated libraries towards dealing with sound manipulations much in the same way OpenGL’s libraries are dedicated towards graphics. It provided an extensive library in which to run and manipulate different sound files in many different ways.

Unfortunately it did require a somewhat extensive configuration process and the vast degree of functionality it provided would not even be required. All that was required was a simple method to play a sound file for a given trigger and the standard “PlaySound()” function proved to be adequate for this purpose. It was incorporated as part of the collision detection and cycled through a number of different sounds files, depending on the trigger.

6.3.5 Aerodynamics
Owing to its complication, correctly modelling aerodynamics with respect to how the tennis ball travelled was always out of the question. It was possible however to create a simple simulation of the effects they might have on it through simple parameter manipulations. For example, the slice shot was altered to have a lower gravitational value affecting it. This simulated the effects that back spin has on a ball, creating a slight upwards force and leading it to retain a straight trajectory for longer. When a ball is given top spin it also tends to dip at a quicker rate. This was observed in the top spin shot through appropriately lessening the power and increasing its initial velocity in the y-direction.

6.4 Tennis Ball Shadow
It was sometimes difficult to distinguish the location of the tennis ball with respect to the court surface. There was little in distinguishing the relative positioning of the ball to the z and y-axes. For this purpose it proved advantageous to introduce a shadow for the ball in helping solve the depth perception issues.

As before, there were a number of ways in which to approach this. The common procedure for rendering of shadows is based on adding a light source and running lighting algorithms to determine the shadows.
For my purposes a simpler solution would suffice. A simple flat circular model was created which posed as the tennis ball’s shadow. When included in the scene, it followed the x and z-locations of the tennis ball while remaining on the court’s surface at all times. This would describe its relative positioning with respect to tennis court depth. To further improve the effect the shadow grew and contracted depending on the current height of the ball mimicking the real life behaviour of a shadow.

Figure 6-1: Tennis ball shadow
Chapter 7: Open Source Community

7.1 Overview

Open source software can prove to be a useful resource in any programmer’s toolkit. It refers to software source code which has been published and made available to the public [20]. Anyone is then free to take and modify this code for their own purposes. A programmer can look to obtain code from an online source when s/he is looking to implement something that has already been accomplished to a full or partial extent.

When used to its full potential, open source code can greatly reduce the coding requirement for the task at hand. This gives the programmer the ability to implement extra functionalities to the program that previously would have been mini-projects in themselves. However, there are also some caveats and disadvantages. The likelihood is that the source code was not constructed with other application in mind and there is a good chance that extensive alterations will have to be made before the code is functional for a different intended purpose. It is also sometimes difficult to assess the reputability of a source and there is no guarantee that the code will function as intended in the first place.

The Open Source Initiative is a corporation that has been set up to solve some of these shortcomings. It is based on a transparency of process and peer review in establishing software standards and regulations to help maintain a higher quality [21]. Efforts such as these continue to add validation to open source software as an effective resource for programmers.

Part of the initial aim of this project included research into the validity of using the open source community in contributing to my project. It was attempted to make use of this resource as often as possible and there were a number of situations where the open source community was found to be particularly useful. There were also a number of instances where seeking to use open source software proved to be more trouble than it was worth.
7.2 OpenGL
From the outset, the proposed method of programming was through OpenGL. OpenGL is the most widely used graphics API in the 2D and 3D graphics industry [22] and proved to be more than adequate for my purposes.

7.3 Models
Improvement of personal artistic and modelling skills was not on the forefront of the project agenda and to that extent it was decided to source the models from online communities. There were a couple of sites with an extended range of 3D models of particular object types but the Google Sketchup model database proved to be especially useful. Google Sketchup was the model editor of choice and it was possible to import any and all objects from this database into the model editor without compromising the quality.

The most important model with respect to the game was the animated hierarchal structure of the tennis player. Based on the structure of the BVH motion capture files it would be necessary to include a skeleton built from 17 different body parts. Sourcing these separate body parts from different sites and expecting them to coincide with one another would have proved to be an improbable task. In order to work around this a model of a man walking was sourced from the Google database and was selected to be chopped up for body parts within the Sketchup editor.

![Man model](image-url)

Figure 7-1: Man model
7.4 Animation files

After conducting some research it soon became apparent that there was a wide variety of animation files to be found online. One such example was the release of a bundle of over 2500 animation files by the website cgspeed.com [23].

It was of great importance to maintain a consistent skeletal structure when loading in the BVH files. These files came in a number of different formats which corresponded to the amount of different locations on the body that were being tracked at a given time. The simple addition of a few extra body parts would have proved to be of little consequence but there also existed some core differences in the structure layout. Different files had different structural paths in which the parent and children bones tended to vary a lot. It would have been necessary to create more than one animation model to compensate for this and would have added a great deal of complication in programming terms. Based on the time it took to create and order one hierarchal model it would take far too long to create a multitude. It was necessary to stick to one structure and therefore the vast majority of online animations proved to be obsolete for my purposes.

The second issue with these files was that there was a severe lack of tennis animations. In the example of 2500 files which were released, there was a multitude animations including basketball, playground, golfing, etc, but none to relating to tennis. Because of this it would be necessary to generate the tennis motions by means of motion capture sessions.

7.5 BVH File Loader

During initial attempts to source a BVH file loader as part of the project the open source community provided a result. A BVH loader in the appropriate C++ coding language had been released by Oshita-Labs [24] and preparations were made for its use and integration.

Unfortunately, this proved to be more trouble than it was worth. To begin with, commenting of the code was done through Japanese and therefore was difficult to understand. On top of this there existed a fair amount of run time issues which caused the program to incur memory allocation errors. An extensive understanding and analysis of the program structure would have been necessary in order to overcome these issues and was perhaps not achievable within the allowable project timeframe. The integration attempt was scrapped and an alternate BVH loader supplied by the graphics department was used.

7.6 GLFont

The existing method for writing text to screen using the OpenGL libraries involved more than a few levels of complication. It was for this reason I decided to use the “GLFont” program that was developed by Brad Fish [25]. It provided the user with an executable that could be used to create a GLFont object. The GLFont methods were then called upon to print out the desired text to screen in a variety of different formats. At first it was necessary to modify certain rendering and viewport settings but once this had been overcome it provided an easy-to-use method of displaying text on screen.
Overall, GlFont contributed greatly to the possibilities made available and proved to be a valuable addition. Text could now be displayed for different purposes such as recording the score and displaying the difficulty.

### 7.7 Push Buttons

The idea behind the addition of mouse functionality was to add to the player’s convenience in controlling the gameplay. One of the most notable options was the “glutAttachMenu()” function which provided the user with a pop-down list of options upon clicking the mouse. Although easy to implement it would also prove too cumbersome a selection method to be used in-game.

It was decided to go with push buttons instead. These buttons would provide an easy interface and quick way in which to change the players shot selection. The method for implementing push buttons was sourced through the open community [26] and greatly reduced the amount of work involved.

### 7.8 Sounds

OpenAL is a sounds library which operates with sounds to much the same extent as OpenGL operates with graphics. It is an extensive library and source of manipulations which can be carried out on sound files. It is released through the open source community and would have been useful in terms of a larger project should a greater range of sound manipulations be necessary.

When seeking to incorporate sounds as part of the project the OpenAL libraries were the first port of call. After some research however, it was found that a simple standard C++ library function call would suffice. It was important to understand the opportunities which were made available through this resource but in this particular circumstance it was not required.

### 7.9 Skybox

There existed an extensive availability of skybox resources in the open software community. The majority of this effort was centred on imagery as artists tried their hands at drawing impressive images. There also existed a number of different tutorials as to how to implement them and other resources which would be needed such as an image loader.

Implementing the skybox proved to be yet another instance of where using open source code can be problematic at the best of times. There were numerous issues about the quality of the code. One such issue occurred in my final and successful attempt. The online site from which the code was sourced [27] provided the source code for the image loader and also an accompanying set of images. The images provided however were not compatible with the image loader, a fact which was not made apparent and this lead to an extensive debugging task on my part. This could have been avoided if a little more forewarning was introduced. However, once the problem was discovered it was easily rectified. Overall the availability of skybox resources greatly reduced the workload which otherwise would have been necessary.
Chapter 8: Evaluation

Once the game had been completed it was time to analyse how it performed. The game was assessed against an industry standard tennis game using a set of standard game reviewing criteria. Lastly the game was given to friends and family in order to gain some feedback and ideas on where there could be room for improvement.

8.1 Game performance

A basic assessment of the game features was conducted and matched up against prior expectations. The overall performance of these features was noted. The game was broken down into separate sections relating to the main areas of focus: Models, animation, realism and gameplay. Each section had various amount of examples of success along with their respective shortcomings.

8.1.1 Gameplay

The tennis game fulfilled the basic task of providing the player with a playable scenario in which it was possible to win and lose. It was a “game-based” gameplay in which the character pitted himself against an NPC opponent and maintained track of his\her progress through the standard tennis points system. There was also a diversification in the number of game play options and ways in which the player could chose to play. These options included control over: aiming, shot selection, character movement, camera angles and opponent difficulty.

AI performance – There remained a couple of bugs with the A.I which caused it to act in an unpredictable manner on occasion. This was due in large part to insufficient measures taken to control the timing of the animation files. This would frequently result in the A.I overshooting the target area on the baseline as it moved to intercept the ball.

8.1.2 Models

The task of introducing models into the game was somewhat a lengthy one but proved to have a certain degree of success in terms of demonstration. An environment of appropriately scaled models was constructed in which textures were used to add to the overall presentation. One area in particular was marked for improvement:

Model Quality – As a result of the 3DS loader being used in the game the models created were required to be of 3DS format which created problems on 2 levels. There was a clear absence of 3DS modelling software and the quality of the models altered in changing from one format to the next. This would have been avoidable with the addition of a 3DS max model loader as part of my project.

8.1.3 Animation

Animation was an area of particular importance and the greatest amount of effort went into making sure that a certain standard of simulated movement was achieved. Of particular note was the integration of motion capture animation files as part of the project. Some of the shortcomings included:

Improper joint correlation - Creating a model structure using a hierarchal structure of bones proved to be troublesome when attempting to fit it altogether in-game. This was principally due to the fact that the pivot points were reset by 3DS studio max on export to the 3DS format. It proved difficult to find these pivot points using the matrix manipulation functions in OpenGL which largely was carried out through guess work.
Lack of animations - It was regrettable in the end that a few more animations had not been available for realism purposes. It was clear that a greater amount of animations would have helped to improve the player movement process. The characters only had one movement available to them to get from A to B and this could only be sped up in the interests of moving faster. A running and jogging animation would have improved the simulation to this extent. This could have been achieved through extra motion capture sessions but sadly this was not possible within the allowable time frame.

8.1.4 Realism
Realism was one area which had proved to be the most limited in the fulfilment of its initial goals. It was necessary to omit some central rules such as momentum. Others, in the cases of gravity and aerodynamics, had been severely doctored for presentation and compatibility purposes.

8.2 Assessment against industry standard
In order to further evaluate the game performance it was decided to perform an assessment of it against a typical industry standard tennis game. The game I chose for this assessment was “TopSpin”. This game was originally released for the Microsoft’s Xbox platform in 2003 [28].

There is no de facto standard way to review a game as every reviewer can have varying sets of assessment criteria. A typical set of evaluation criteria was sourced which attempted to review all components of the games in order to get the most comprehensive results [29]. Each one of the core performance areas was given a mark from 1-7. The totals were added up and an overall mark was given for each. A breakdown the review criteria and of the results is included in Appendix A.

It was evident that the relative performance of the tennis simulator suffered heavily due to the bug in the NPC’s ability to return the serve. This was identified as the main contributing factor to its low performance against Top Spin. It was obvious that rather large differences were present in areas such as graphics and animation. Although it was lacking in these areas it would have proved a great deal more trouble to pick up extra marks in these areas owing to the large amount of effort required in integrating a proper graphics element.
Also of note was the relative similarity between other ratings in the games. Features such as gameplay opportunities and program flow proved to be quite comparable to one another between both games. This perhaps can give an explanation for the successes observed by small scale game productions within the industry. Relatively small scale productions can still go on to be very popular and continue to prove that small game productions are a viable option. This has been proved on multiple accounts by single person productions such as “Desktop Tower Defence”, designed by Paul Preece and as of July 2007 has been played 15.7 million times [30] and “Minecraft” developed by Markus Persson, released in 2011 and has made €23 million in revenue [31].

8.3 Feedback
The game was evaluated by a group of 14 friends and family, all of whom were asked to be honest in their evaluations. A survey was composed with a few simple questions to get a rough idea of some of the main areas of focus. The results of this survey are included in appendix B.

The overall results seemed to compliment the rating that resulted from the industry assessment. Certain key areas such as overall fun and challenge presented were severely lacking. This was to be expected from the lack of consistency brought forth by the A.I.

The visual aesthetics and in game physics also received stunted reviews. This can be somewhat explained by the nature of graphics and realism being part of the large budget production realm.

As a slight contrast to this the gameplay aspects did receive more positive feedback. There seemed to be ample gameplay options present to be able to mix up the play to some extent and a tolerable means of accomplishing it (controls).
Chapter 9: Conclusion

9.1 Project Goals
The primary goal of this project was to create a player controlled tennis simulator and to that extent it was successful. A game was produced with a certain standard of realism which was supported multiple by gameplay options. Feedback from the game, although not wholly positive, did point out some areas in which improvement was needed.

9.1.1 Game implementation
The core features of the game were analysed as to the degree in which they had been accomplished:

A successful pipeline for model integration had been indentified although several areas of improvement were identified. A lack of integration of more complex model formats such as type 3DS max was a key area which could have been built upon.

Gameplay proved to be one of the more successful implementations of the simulator as was confirmed by the feedback. A number of gameplay options were made available to the player and an appropriate level of thought had been put into the A.I., which although problematic, still succeeded in providing a certain degree of challenge.

Character animation for both players was made possible through the creation of a skeletal object which stored animated sequence data. The use of motion capture technology proved instrumental in adding to the animated effects. The degree of incorporated animation was appreciated by the testing group as the overall satisfaction proved to be quite high.

The game physics were not implemented to the initial desired degree and had been heavily modified from their appropriate real world values and models in order to ensure proper game flow. However the resulting simulation of physics proved to be somewhat of a satisfactory alternative.

9.1.2 Using Open Source
The use of the open source community proved to be a very valuable resource in contributing to various features throughout the project. It would not have been possible to code all the various areas without the support of existing code. Certain problems within this process were highlighted such as code incompatibility, re-configuration and questionable standards.

9.1.3 Industry comparison
The comparison of the simulator with the industry standard highlighted some areas in which the lower scale production could not compete with the larger. Areas such as model creation and animation proved quite work intensive and required a large amount of time and effort to get right. It became apparent that it was possible to implement some aspects to a certain degree of competency but to go beyond that would exponentially escalate the work load required.

It was also worth noting that certain aspects of the game were of greater similarity in their performance. This suggested that game design is not primarily influenced by computationally demanding processes such as detailed graphics and complex animations. There are plenty examples of lower production classics such as Super Mario Bros for the NES and Pokémon for the Gameboy that still are a testament to the importance of core gameplay values.
9.2 Future Work

“Tennis simulator” would be very transferable across a number of platforms due to the large reliance on open community software. In this way it could be made available to a wide variety of devices. Integration as part of the Trinity Metropolis project could also be an achievable outcome.

9.2.1 Improving the simulation

Although limited in feature scope, the tennis simulator could serve as a template for future means of game creation of similar specification. Some of the processes which were involved are standard across all platforms and certain software applications were identified as being particularly useful in the games construction. A number of areas were earmarked for improvement and when worked upon would greatly add to the basic functionality of the game.

9.2.2 Industry Analysis of Gaming

The subject of the criteria that are most important to the gaming industry was touched upon. It has been interesting to see how some of the older less funded productions have held their own against the newer Titans. Adding on this, it could prove to be beneficial to do an in depth study of what players appreciate the most within a game. This could help major game studios to allocate their resources in a more appropriate manner when it came to improving key performance areas. This would result in the production of a better rounded game and save expense in areas of little payoffs.

9.2.3 Open software support

The existence of the open software community is rarely viewed in a negative light. The resources that are available at the fingertips of the typical programmer are immense. If a certain application or functionality has to be coded, chances are it has been done before and possibly available for download online.

Over the due course of this project certain disadvantages of the open source community were experienced firsthand. There are already existing bodies such as the Open Source Initiative which help to maintain certain coding standards. However, there is a lot of potential for presentation of the available resources in a more intelligent and intuitive manner.

Further study could be conducted into grouping resources together in a much more relational manner for the particular programmer’s task requirements. The programmer would have instant access to key information such as: key software tools, the means of approaching the problem and coding examples of how it had been done previously. For example, a low level gamer starting out looking to implement a game in C++ using OpenGL would have instant access to a database of knowledge which related to this specific task.

Creating structures such as this might prove to be a very great task indeed and borders on the issue of a semantic web. However, once implemented, it could offer numerous benefits over the current system in place... the browser search bar.
Bibliography


2012].


### Appendix A: Industry Comparison

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clarity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Spin</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flow</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Spin</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Balance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Spin</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Spin</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Integration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Spin</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fun</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Game</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Spin</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Clarity**

Game: 5 – The layout was easy to understand and there was little ambiguity.

Top Spin: 6 – There was a greater use of visual aids and sounds to draw the players attention to what was happening next.

**Flow**

Game: 6 – The game proved to be very streamlined and had little rule exceptions or cumbersome exceptions.

Top Spin: 6 - Equally as Streamlined.

**Balance**

Game: 3 – The game was balanced heavily in favour of the player regardless of the skill level chosen. This was due to faults in the A.I. which made it repeatedly miss the target.

Top Spin: 7 – It was evident that a great deal and effort had gone into providing a balanced experience within the options of different skill levels.
**Length**

Game: 4 – Although the game is equally as long as Top Spin it doesn’t have as much to offer in terms of gameplay. This means it becomes repetitive after a short while.

Top Spin: 6 – This game provides plenty of opportunity for keeping the player engaged and interested throughout the entire duration of a game.

**Integration**

Game: 4 – The character abilities and controls strongly compliment the degrees of freedom that are available within a tennis match. Having said that there is a significant lack of them but the ball dynamics do simulate somewhat realistic behaviour.

Top Spin: 6 – There is a greater deal of balance and thought that goes into each gameplay option that is available to the player. It all comes together with the game physics to provide a much more believable experience.

**Fun**

Game: 3 – The element of fun in this game is largely hampered by the NPC’s ability to return the ball in a consistent manner. It is rare that a point is fought for over a lengthy rally and because of this it becomes hard to become as involved.

Top Spin: 6 – The Gameplay is an enjoyable experience. It is possible to provide an appropriate and consistent challenge for the player depending on the skill level that is chosen.

**Scores:**

Tennis Simulator 25/49

Top Spin 39/49
Appendix B: User Feedback

Rate your overall enjoyment

Did you find it challenging?

Rate the Visual Aesthetics

Rate the game physics

Was there an adequate amount of gameplay features?
Appendix C: Project Code
Please find attached a copy of the project source code.