UNIVERSITY OF HERTFORDSHIRE

Faculty of Engineering & Information Science

BACHELOR OF ENGINEERING/SCIENCE
DEGREE/DEGREE WITH HONOURS
IN MULTIMEDIA TECHNOLOGY

PROJECT REPORT

3D MAZE COMPUTER GAME USING JAVA 3D

Anish Shah
April 2004
Abstract

This report gives detailed documentation of the work carried out to create a 3D maze computer game using Java 3D and the importance of project planning as well as the design approach used. Java 3D is a set of libraries of Java that offers many graphical features for developing applications. As gaming becomes more popular day by day, it has come to a point where new boundaries need to be explored and Java 3D is a starting point. Using different aspects of Java 3D such as lighting and texturing as well as the more detailed area like movement in first person view, a game can easily be formed. The conclusion describes the analysis of work accomplished and if it was a success.
Acknowledgements

I would like to thank all the people involved in helping and supporting me throughout this project. A special acknowledgement goes to my supervisor Dr. Munther Gdeisat for his time, support and enthusiasm on assisting me during the time of this project.

I would also like to thank my friends and family on guiding me in the right direction, when times were doubtful. A final thanks goes to Felix Zhu for his dedication to final year students.
Contents

Abstract II
Acknowledgements III
Contents IV
Glossary of Terms VII
List of Diagrams IX

1 INTRODUCTION 1
  1.1 Reasons for the Project 1
  1.2 Project Aims 1
  1.3 Project Objectives 1
  1.4 Restrictions 2
  1.5 Author’s Background 2
  1.6 Structure of Report 2

2 BACKGROUND INFORMATION 4
  2.1 Introduction 4
  2.2 Basics of Java 4
  2.3 What is Java 3D? 4
    2.3.1 Java 3D Goals 5
    2.3.2 Strengths 5
    2.3.3 Weaknesses 5
    2.3.4 Java 3D in Games 6
  2.4 History of Mazes 6
  2.5 Research of Maze Games in Websites 6
  2.6 Investigate Human Interaction with Computers 9
  2.7 Summary 9

3 ANALYSIS AND REQUIREMENTS 11
  3.1 Introduction 11
  3.2 Requirement Analysis (User Requirements) 11
  3.3 Requirement Analysis (Technical Requirements) 11
    3.3.1 TextPad 11
    3.3.2 Macromedia Dreamweaver 4 12
    3.3.3 Adobe Photoshop 6 12
    3.3.4 Java 3D 12
    3.3.5 HTML 13
    3.3.6 Hardware 13
    3.3.7 Alternative Software 14
  3.4 Summary 14

4 PROGRAM OF WORK 15
  4.1 Introduction 15
7.3 Analysis of Project 39
7.4 Project Planning 41
7.5 Alternative Methods 41
7.6 Further Development 42
7.7 Summary 42

8 REFERENCES 43

9 BIBLIOGRAPHY 44

APPENDICES 45
Appendix A – Gantt Chart 46
Appendix B – Full Source Code 49
Appendix C – Maze Diagram 61
Appendix D – Questionnaire and Results 63
Appendix E – Screenshots of Game 66
Glossary of Terms

API
Stands for Application Programming Interface. The interface by which an application program accesses the operating system and other applications.

Applet
A small program that can be sent across a network and interpreted safely on the receiving machine.

Classes
These are defined as declaring variables and functions associated with the objects of that class and also with the class itself.

Compiler
A program that reads source code, translates it into machine language, which then writes it into binary code for it to be executed.

Gantt chart
A graphical representation of the main stages or activities in a project work plan over time.

GUI
Stands for Graphical User Interface. A technique involved in using graphics, along with a keyboard and a mouse, to provide an easy-to-use interface to some program.

HTML
Stands for Hyper Text Mark-up Language. It is a simple language for displaying content on the World-Wide-Web.

import
A Java programming language keyword used at the beginning of a source file that can specify classes or entire packages to be referred to later.

Internet
The Internet is a super highway network. It allows computers to exchange information by having small networks interconnecting with each other.
**Java SDK**
The Software Development Kit (SDK) is a development environment for building applications and applets using the Java programming language.

**Java Development Kit (JDK)**
A software development environment for writing applets and applications in the Java programming language.

**Java Virtual Machine**
The part of the Java runtime environment (JRE) responsible for interpreting byte codes that can be used on various computer platforms.

**JPEG**
Stands for Joint Photographic Experts Group. A graphic image file or an image compression algorithm.

**OpenGL**
OpenGL is a graphics software development kit for PCs and Macs used to develop 3D games that is written in the C programming language.

**public**
A Java programming language keyword used in a method or variable declaration. It signifies that the method or variable can be accessed by elements residing in other classes.

**Tags**
These are formatting codes used in HTML documents. Tags indicate how parts of a document will appear when displayed by a browser such as Microsoft Internet Explorer.

**URL**
The World Wide Web address of a site on the Internet.

**VRML**
Stands for Virtual Reality Modelling Language. It is a graphical system that allows creating 3D images. It is used for environments for the purpose of games and education.

**WWW**
Stands for World Wide Web.
List of Diagrams

Figure 2.1: http://dev.javaboutique.internet.com/3DMaze. 7
Figure 2.2: http://amadeus.uprm.edu/~goopm/MainProg.html. 8
Figure 3.1: A simple rotated cube. 13
Figure 4.1: Waterfall methodology. 18
Figure 4.2: JAD methodology. 18
Figure 5.1: A right handed co-ordinate system. 21
Figure 5.2: Defining centre point of a box. 22
Figure 5.3: The scene graph structure. 23
Figure 5.4: A simple maze design. 25
Figure 5.5: Wrong co-ordinate placing. 26
Figure 5.6: Right co-ordinate placing. 26
Figure 5.7: The starting post using text 2D. 30
Figure 5.8: Final game created. 31
Figure 5.9: How JVM works. 32
Figure 5.10: Layout of webpage. 33
Figure 7.1: Specification of computers tested with 3D maze game. 39
Figure 7.2: Game with no back wall. 40
Figure 7.3: Game with back wall. 40
1 INTRODUCTION

Multimedia is a collection of texts, images, sounds and interaction on a platform. These can be integrated together to create a game that can be played on any PC, that has sufficient memory.

Evolution in technology has provided everyone opportunities for exploring and interacting with 3D worlds, which not only is it for amusement and enjoyment but it enhances multimedia applications to gain awareness for potential clients and customers, which can bring good fortunes to a business.

1.1 Reasons for the Project

Java is well known in creating applications, but many people do not know about the capabilities it has on the 3D environment. The creation of the game brings interests of what Java 3D can do. Games like Quake and Doom have brought huge profits to the creators, and this can be achieved in Java 3D with a little learning and patience. The Internet is already the fastest medium that people have started to use, which means that users of the Internet are always on the prowl for something new. Java 3D can be incorporated onto a website, which can result in recognition of that website.

1.2 Project Aims

The aim of this project is to create a maze game using Java 3D, by researching and learning the programming language. The game will demonstrate the benefits of Java 3D and what it can accomplish. Using various aspects of Java 3D such as lighting and texturing the game will have an appealing curiosity to it, which will draw people of all ages to play the game.

1.3 Project Objectives

The objectives of this project are:

- To create a full working source code of the game.
- To create a user friendly webpage for the game.
- Making the game appealing to any player.
- To make the webpage and application available to the author, so once the project is complete and after its deadline it can still continue as an ongoing project.
- To create a game that can be played on even the lowest specification computer.
- To create the game in first person view.
- To create the games using primitive shapes.
1.4 Restrictions

While working on the project, there were a number of restrictions that delayed the movement of the game. These setbacks included:

- To create such a game, a suitable graphics card needs to be installed. A minimum specification of 32MB 3D card is required, with Direct X capability.
- To get in touch with people that have experience working with Java 3D.

1.5 Author’s Background

The author has studied Multimedia Technology at the University of Hertfordshire for the last few years, and prior to that HND Computing with Software Engineering. From which both courses the author has built up very basic knowledge of Java to create a multimedia application. However the author has no knowledge of Java 3D, which is a requirement to successfully meet the aims and objectives of this project. This can be considered a restriction to the project, but the author can overcome this by becoming familiar to the language.

The author has many years of experience of different types of games on different types of platforms. This has enabled him to have a broad mind on gaming, and the way its best played to get the most out of them.

1.6 Structure of Report

The report has been written for two types of readers, the first of the readers is being an experienced technologist who is not a specialist in the field and the second being a technologist who is an experienced in this field of the project. Readers that are experienced in this subject may wish to skip the initial chapters to gain rapid conclusions that the author has made. Readers with brief or no knowledge of the subject are advised to read all chapters to gain understanding of the entire project.

The report consists of seven chapters. A brief description of each chapter is outlined below.

Chapter 1 – Introduction

This chapter provides an introduction to the entire project, the restrictions, the author’s background as well as required aims and objectives.
Chapter 2 – Background Information
Chapter 2 provides background information to readers that are not familiar with the concepts of Java and Java 3D, and provides useful information on mazes and its history.

Chapter 3 – Analysis and Requirements
Chapter 3 provides the user requirements for the project. Technical requirements are also included in this chapter, as well as an overview of the software and hardware needed to complete such the project.

Chapter 4 – Program of Work
This chapter confers how the project was managed, the project methodologies and the importance of time management.

Chapter 5 – Design Approach
Chapter 5 is of a more technical area of the report. This is where jargons are expressed, when the project was going through work stages. It also covers the problems encountered and how they were solved.

Chapter 6 – Testing
With any application testing is a must to achieve the aims and objectives detailed. This chapter will look at two types of testing and how they are adopted to this specific project.

Chapter 7 – Conclusion
This chapter discusses how the aims and objectives are met and an overview of the project. It also looks at alternative methods, further developments and project planning.
2 BACKGROUND INFORMATION

2.1 Introduction

Background information is vital before commencing on any projects, even outside the digital world as it forms a foundation to completing the end product. This chapter is useful for readers that are rusty with the concepts of Java and Java 3D or readers that have no understanding of Java. It should be noted that Java is a very broad subject and it cannot be practical to include all aspects of Java. It also looks into the history of mazes, Human Computer Interaction (HCI) and concludes with a summary.

2.2 Basics of Java

Java is a programming language that was created as a small project by a company called Sun Microsystems. It is now known to be a widely used programming language. The significance of this has resulted in Java being used in applications ranging from large to small in any type of business. It is also used for simple web based games, which can easily be found using a search engine.

Java lets you write powerful programs that can run from browsers, from any desktops, on a server, or any compatible devices such as mobile phones. The smart thing about Java is that it can be run from any machine, with a program called ‘Java Virtual Machine (JVM)’. This means that a machine that has never seen Java before can run any specific program that was created on another machine remotely away.

There has been many extensions to the core Java to increase popularity, one of them being Java 3D. There have also been extensions for 2D graphics and Imaging API’s.

Java is commonly used in applications and applets. Applications are described as being standalone programs, whereas applets being similar to applications do not run as a standalone application. Applets follow some sets of rules that allow them to run on compatible browsers, such as Microsoft Explorer and Netscape.

2.3 What is Java 3D?

Java 3D is an Application Programming Interface (API) used for writing three dimensional graphics in applications and applets. It is a library of the core Java.
Java 3D class library provides a simpler interface than most other graphic libraries, but still has the capability to produce excellent animation and of course games. Java 3D uses existing technologies such as Direct X, to improve the processing of any type of games, with the help of a good graphics card. This helps to reduce slowness and improves smoothness.

2.3.1 Java 3D Goals

Java 3D was designed with several goals in mind:

- To deliver the highest level of performance to application users.
- To provide a rich set of features for creating interesting 3D worlds.
- To provide high-level object-oriented programming ideas.
- To provide support for other types of 3D programming graphics, such as VRML.

2.3.2 Strengths

Java 3D can provide programmers with strong points, these are:

- Java 3D provides a high-level, object-oriented view of 3D graphics. This is accomplished by using a scene graph-based 3D graphics model. This allows scenes to be easily described, transformed and reused.
- Java 3D is optimized for speed where possible, by using the on board processor on a graphics card.
- A large and growing number of 3D loaders are available to import content into the Java 3D runtime, as mentioned before VRML.
- It allows users to program entirely in Java.
- Able to ‘Write-Once-Run-Anywhere’.

2.3.3 Weaknesses

With strengths usually come weaknesses, these are:

- Java 3D is a standard extension API, which leads to most programmers having to struggle to keep up with changes and additions to the core platform.
- Compared to OpenGL which is available for every type of operating systems, the cross-platform portability of Java 3D has many problems.
Distributing Java 3D to the client side can cause a bit of a stir, as the 3D libraries need to be installed on that machine.

2.3.4 Java 3D in Games

Programmers of any type of 3D game have used hardware to its optimum. In the past developers have tended to stay away from easy-to-use software such as Java 3D. However 3D games today tend to over power 3D hardware accelerators such as a graphics card and to use less tricks in rendering.

Java 3D was not clearly designed to match the game developers expectations, but the sophisticated implementation techniques provide more than enough performance to support many games, especially designed for the use over the Internet.

2.4 History of Mazes

Maze is the English word for a labyrinth, technically it is lots of paths, leading to one centre or a way out although there are hedges in between, and the player gets lost in it. They can be as simple as a game in a child's colouring book, or even in as a house of mirrors. They can also be entertainment when travelling e.g. by aeroplane or even a path along a deep spiritual journey.

"It is a confusing, intricate network of winding pathways; specifically with one or more blind alleys".

In our digital world mazes can be found as a game on a computer. There are many websites that allows you to play in a maze.

2.5 Research of Maze Games in Websites

Various websites were investigated to find different types of mazes available on the Internet. All these websites were critically analysed. From this analysis, aspects that are required to produce a successful project are highlighted.

1 The UnMuseum – The History of Mazes http://www.unmuseum.org/maze.htm

The first website analysed shows a very complex maze. It is on a website that shows what Java can do. The language to create this maze is not in Java 3D. The creator of this maze is James L. Dean. This maze can be found on numerous other websites. Figure 2.1 shows the design of the maze.

Good points of the maze:

- More time spent trying to find a way out.
- Can form a new maze, with just one button click.
- The maze can be solved if frustrated and cannot find a way out.

Bad points of the maze:

- Can get frustrating to find a way out.
- Not in first person view.
- Visually it’s very tedious to look at.
- No sound.

2. http://amadeus.uprm.edu/~goopm/MainProg.html

The second maze on a website that was selected can be seen on Figure 2.2. This maze was selected as it illustrates a game in first person view.
Good points of the maze:

- The maze is kept on a very simple webpage.
- In first person view.

Bad points of the maze:

- Takes an exceptionally long time to load up.
- Colours are too bright, not good for the player’s eyes.
- No sound.
- Applet size too small.
- Uses the numeric keypad to navigate through the maze.

![Java RayCaster Example](http://amadeus.uprm.edu/~goopm/MainProg.html)

**Figure 2.2:** [http://amadeus.uprm.edu/~goopm/MainProg.html](http://amadeus.uprm.edu/~goopm/MainProg.html).

Overall the first website was more exciting to play, but did get a bit frustrating. It showed very excellent programming to construct a new maze with just one button click. The downside of this maze was that it was not visually appealing. The second website was more appalling as it had all the bad attributes from the first website. One of them being poorly chosen colours which strain a player’s eyes, it also uses the numeric keypad which makes it difficult for a player to navigate around the maze, compared to the conventional arrow keys. The upside of this website was that it is in first person view, it is very straightforward, and does not have jargons, animations or sponsors on any part of the website.

The analysis from the two mazes and their websites have produced some user requirements. To have a successful game on a website, it must comprise of the following points:

- To make it visually appealing, this can include textures and lighting effects.
To have sound, to give it the gaming environment.
To have the website simple, but yet inspiring.
To have a speedy load up time, for players with a slow Internet connection.
To make it a first person view game.

2.6 Investigate Human Interaction with Computers

"Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them." ²

The term Human Computer Interaction (HCI) is when a person or in this case a player interacts with graphics and animation to manipulate surroundings. Getting the results for HCI is important in constructing a game on a webpage. The results could mean whether the game is successful or not. With HCI everything is useful, such as computer graphics, development environment and even the operating system to human aspects such as viewing colours, interaction with an input device to even headaches (playing a game for a long period of time).

So many things need to be taken into consideration when creating a successful game in a webpage, such as 'What will make the player want to play again? There are so many features to be included, but the most important aspect is to create the game with unique content. Making the whole environment of a game unique, has many excellent benefits such as that it provides self pride in such a creation and will make the player want to revisit time and time again. The downside of having a unique gaming environment is constantly updating it. A player needs to see different visuals, different content every time he or she revisits the site to play the game.

The appearance of the site is important, as well as the gaming features such as texturing, but so is speed. Consideration needs to be applied to players that do not have very powerful computers, nor very fast download transmission. If it takes too long to play the game then the player will be fed up of waiting and not play the game.

2.7 Summary

It is clear to see what is required of a 3D maze computer game by looking at this chapter. Looking at different maze games on the Internet gives the impression the something unique has

² Human Computer Interaction: http://sigchi.org/cdg/cdg2.html#2_1
to come out of this project, something that will want players to play over and over again. Therefore the aim of this project is to fill that void.
3 ANALYSIS AND REQUIREMENTS

3.1 Introduction

This chapter focuses on what is required by the user and the technical requirements for creating the 3D maze game. It also includes what software and programming languages are required to create such a game and the webpage.

3.2 Requirement Analysis (User Requirements)

The aim of all projects is to give what the user wants, a user is what makes something remarkable, without the user there would be no point in doing anything. Accumulating all necessary information will lead to fulfilling these requirements. Using the analysis from Chapter 2 the following user requirements have been made:

- To make the game visually appealing, this can include textures and lighting effects.
- To include sound, to give it the gaming environment.
- To have the website simple, but yet inspiring.
- To have a speedy load up time, for players with a slow Internet connection.
- To make it a first person view game.
- To make the whole process of playing simple yet effective and fun.

3.3 Requirement Analysis (Technical Requirements)

To meet the aims and objectives of this project the following software, programming languages and hardware are required:

3.3.1 TextPad

TextPad is a powerful, general purpose editor for plain text files. Easy to use, with all the features a user requires. It is very popular because it has been integrated with Windows XP, which it makes it look and feel simple to use. It is ideal for integrating compilers, such as Java JDK.
3.3.2 Macromedia Dreamweaver 4

Macromedia Dreamweaver is a program that creates professional websites and webpage’s, as well as designing them and maintaining them. It can be used by professionals or even novice users, with a little experiment with the program. As with most webpage’s everything has a place, such as headings should be placed at the top, and main body in the middle, however this can vary but generally it tends to be this way. Dreamweaver lets the user do this, by having features like layers, tables and frames. With these features the website could look very attractive and have a very appealing view to it. With Dreamweaver it uses the language HTML to create these webpage’s, so the user does not need to write the HTML coding from scratch.

3.3.3 Adobe Photoshop 6

Adobe Photoshop allows the user to create and manipulate images and photos. This is a very powerful package that is used across many businesses and institutions to get most out of just images and photographs. It has many features such as:

- Layering effects.
- Special effect filters.
- Painting tools that allow replication of media.
- A standard pen tool that allows drawing accurate graphics.

3.3.4 Java 3D

As mentioned in Chapter 2 Java 3D is a library of Java. To understand Java 3D it is advisable to learn the basics of Java first. Java 3D allows to program 3D graphics in a structured way, by this the coding is at a distinctive place rather then another type of programming language such as VRML. Figure 3.1 shows a simple example of what Java 3D can do:
3.3.5 HTML

HTML stands for Hyper Text Mark-up Language, the coding language used to create Hypertext documents for use on the World Wide Web. HTML looks a lot like old-fashioned typesetting code, where you surround a block of text with codes that indicate how it should appear. In HTML you can specify that a block of text, or a word, is linked to another file on the Internet. HTML files are meant to be viewed using Web browsers, such as Internet Explorer or Netscape.

A few essential HTML tags are described below:

<HTML></HTML> - This is the opening and closing tags for web browsers.

<HEAD></HEAD> - Anything inside this tag is not viewed on the browser.

<BODY></BODY> - Anything inside this tag, which normally is text and images is viewed on the browser.

3.3.6 Hardware

Each software has its own specification of what is required from a workstation. What is gathered here is what is required of the specification with all the software on just one workstation:

- 1GHz Intel Pentium or equivalent.
- 256MB of RAM.

Figure 3.1: A simple rotated cube
- Modem – 56k or Higher.
- 2GB of storage space of the installation of all software.
- 32MB Graphics card with Direct X capability.

3.3.7 Alternative Software

To create the game no alternative software is needed, as the project is to create the game using Java 3D, therefore only Java 3D will be used as a programming language. Microsoft FrontPage can be used to create the webpage instead of Macromedia Dreamweaver. FrontPage is similar to Dreamweaver as it has the same attributes, such as layers, tables and frames. It is part of the Microsoft Office package so we know that it is user-friendly. The advantage it has over Dreamweaver is that it is straightforward, although it lacks the professionalism that Dreamweaver has.

3.4 Summary

This chapter has focused on what is required to create a 3D maze computer game. The user requirements have been defined, as well as the technical aspects needed. The necessary software is outlined, with a brief of what it can do. The hardware noted is the minimum specification that is required, although a more powerful workstation the better.
4 PROGRAM OF WORK

4.1 Introduction

This chapter will concentrate on how the project was controlled. Decision making, making sure the project goes to plan and meeting the aims and objectives are a must if a project is to be successful. Time management is the far most important aspect of a project, and it is detailed in this chapter. Also included are the different design approaches that were considered, and the one that was applied to this project.

4.2 Project Management

"The success of a project will depend critically upon the effort, care and skill you apply in its initial planning." ³

As with any project, may it be software or labour work managing time is essential. Going according to the scheduled time to complete a task makes the project a complete success. One of the ways that this can be accomplished is by using a Gantt chart.

The project was broken down into manageable tasks with time deadlines. Each task was estimated on how long it would take to complete, so that the next task could get underway. The reason for this was to monitor any point of the project duration, so that it could be said that if a task was taking too long, then a solution should come too effect to resolve this setback. Some of the tasks that were assigned to the project are listed below:

- Research into Java 3D.
- Work on existing examples.
- Create the game – This task was split into smaller tasks, such as:
  - Create a primitive shape to represent a wall.
  - Add lighting to the scene.
  - Add sound.
- Test the software.

³ Planning A Project, Gerard M Blair: http://oldee.see.ed.ac.uk/~gerard/Management/art8.html
Many more tasks were added, such as creating the webpage. The Gantt chart illustrating the project stages can be found in Appendix A.

There are seven stages that have been followed for the project to be a success:

4.2.1 Feasibility Study

Feasibility study is a combination of a market study and an economic analysis that provides a client with knowledge of both the environment where a project exists and the expected return of what the project can achieve. The feasibility report had details of hardware and software requirements, to complete this project.

4.2.2 System Design

This section is where all the coding of the project was assigned to. It composes of sub sections, which are broken down tasks that are allocated specifically. This formulates the process to be efficient and no task was started until the previous one was complete, unless it could not be done at all.

4.2.3 Presentation Seminar

This seminar was provided to ensure that the project was going on the right track, to get a satisfactory conclusion. It helped the author to express ideas orally with clarity and persuasiveness. The seminar covered what was detailed in the feasibility study.

4.2.4 Testing

With any software developing, it is vital that testing plays a major part to the project. If a problem or error arose then it was solved and then retested. Testing was completed in only a couple of weeks, as the project had a deadline.

4.2.5 Project Demonstration

After all project work was completed, a demonstration was followed. Here the top supervisors, or in this case were the clients, viewed the project to get an understanding of what was achieved.
4.2.6 Project Report

This is an in-depth report that takes into account every little thing that happened during the project work, including setbacks.

4.2.7 Poster Session

The final stage of the project work is the poster session. Here a report is concluded of the project as a whole. This is a summary of the project report, outlined in section 4.2.6.

4.3 Design Methodologies

The first step in creating a project, is deciding which methodologies are going to be used. There are many methodologies that can be used for either creating a software or designing a website. Although there are other methods that were considered and deciding which one to use can be of a difficult task, as there are always certain amounts of creativity involved and evaluating between two correct methodologies. When there is only one programmer creating a project, then it is up to the programmer to see which one he or she is confident with, to achieve maximum results. The methodologies that were considered are the Waterfall and JAD models.

4.3.1 Waterfall Method

The Waterfall method is defined by starting one task and then finishing it before heading towards the next task. A later stage may reveal that extra work may need to be done at an earlier stage. Stages can include requirement analysis, to system design, to creating the program, to testing and so on. An advantage to this method is that a programmer works on one task which can give ideas for the following tasks. An alternative description is that it gives the programmer the flexibility to plan ahead even before the project gets going. Figure 4.1 shows this method as a diagram.
4.3.2 JAD Method

This method works with being close to clients and working with them process after process. The steps that are followed are to create a webpage or application or anything similar and refer back to the client and get feedback on, if this is what they require. If it isn’t and changes need to be made then it’s back to the drawing board, and then the process is then repeated, until the client is satisfied with the final product.

Figure 4.2: JAD methodology.
4.4 Project Methodology

After careful consideration, the methodology that thought to be the suitable approach had to be the Waterfall method. The reason why this method was chosen was because the author had confidence, from previous project work. As this project was to create a game, software coding had to be tested throughout the process of coding and the Waterfall method tailed to this need.

4.5 Summary

This chapter has looked the approach that was taken into completing the project. The main point that was raised in this chapter was the importance of time management and detailed the advantages of using a Gantt chart. The methodologies that were considered were analysed, and the chosen one was outlined.
5 DESIGN APPROACH

5.1 Introduction

The aim of this project was to create a 3D maze game in first person view using Java 3D. To achieve this many steps had to be taken, such as learning the way 3D graphics are programmed in Java. The main achievement in the game was to make the player feel that he or she has enjoyed playing the game.

To meet the aims and objectives it is critical that the project title had to be fulfilled. The game will be played over the Internet as a means of stress relieving to workers, students or just web surfers.

This chapter will view what action was taken to create the game and the webpage associated to it.

5.2 The Principles of 3D Graphics

The aim of this project is to create a 3D maze game, so in this chapter it is appropriate for the reader to understand the principles of 3D graphics. 3D graphics can be said to be just like 2D but with depth.

Java 3D follows a right-handed co-ordinate system. There are other 3D packages that follow this way too, such as VRML. Without getting too technical, defining co-ordinates is to do with mathematics (X, Y, and Z).

- X refers to the width of an object
- Y refers to the height of an object
- Z refers to the depth of an object.

Let’s understand the right-handed co-ordinate system a little further.

1. Hold out your right hand with your thumb pointing right.
2. Point your index finger up.
3. Point your middle finger towards you.
If you’re a little confused then take a look at Figure 5.1.

![Figure 5.1: A right handed co-ordinate system.](image)

The thumb represents the X axis, the index finger represents the Y axis and the middle finger represents the Z axis. Where all three meet, is the centre point (0, 0, 0). If this is to be implemented in the programming part then it is essential that it is understood even further. The walls that are created in the game are programmed as primitive boxes. Each box (wall) is defined in the code as different sizes. Let’s look at the box in Figure 5.2. The height of the box is defined as 1, the width of the box is defined as 1 and the depth of the box is defined as 1. The result of this is a perfectly square box. Here is the code to show how the first box is defined in size in the source code of the game:

```java
Box Wall1 = new Box(1f,1f,30f,app1);
```

The above code is at a height of 1, width at 1 and depth at 30. Placing the box (wall) at a specific place in the scene is discussed later in this chapter.
5.3 How to Code?

Java 3D was programmed to work along side with TextPad. The way it works was that coding was typed into the editor and then there was a function to compile the code, if the code did not compile then the editor would throw back errors. The errors would specify the requirement for the programming language to work. The code was re-worked on and then re-compiled. When there were no errors, the game would run.

5.4 The Essentials of Java 3D

Creating this game involves a structured type of programming, in other words all coding that is programmed needs to be put into some sort of order. The way this is achieved is by creating a scene graph. Within the scene graph are nodes, the nodes that are used in the game are labelled as the content branch and the view branch. This is where things get a little more detailed.

5.4.1 Scene Graph

A scene graph is a structure that holds the information within the scene; these include the geometric data, the appearance definition and the viewing parameters as well as loads of other things. The size of a wall described in section 5.2 is an example. Figure 5.3 shows the scene graph used to create the project.
At the top of the diagram is called the VirtualUniverse, this is consider the root node and is an instance of a class. Every program constructed using Java 3D usually has one of these, in this instance it does have just one. The rest of the scene information is referenced by the VirtualUniverse as Locale objects. A Locale node denotes a location in the VirtualUniverse at which point the rest of the scene content subsists. There can be more than one Locale node in a program, but in this project only one exits. The VirtualUniverse in the diagram shows that it is a parent of the Locale node although this is not always the case. From the Locale there are two BranchGroup nodes: Content branch which stores the scene content and View branch which stores the definition of the view parameters.

### 5.4.2 Content Branch

The Content Branch contains the actual 3D content to be displayed in the scene, such as the box primitive shape described earlier in this chapter. Other elements are stored in this node such as the sound node; this will be discussed later in the chapter. The TransformGroup allows geometry shapes to be positioned in the game. The Shape3D node is where the geometric information in the game exits. There are many Shape3D nodes in the coding as it represents different shapes, such as numerous walls.
5.4.3 View Branch

The View Branch positions how the scene will be looked at and it also defines the way it is displayed. In the project the BranchGroup node is the first one to be referenced, and then it is the TransformGroup. The TransformGroup positions the view of the game in the Universe. It does this by storing a transformation that affects nodes such as the translation of the Y axis. The final node in the View Branch is the ViewPlatform; this defines the co-ordinates to be viewed.

5.5 Game Design

Designing a game can lead to questions on how and where to start, this can be frustrating and time wasting. With a bit of planning this can be resolved. The first thing that needs to be considered is what type of game will be created, obviously as the title of this project is ‘3D maze computer game using Java 3D, it is already formulated. The game that is created is a maze game and the language that will be used is Java 3D. Creating a maze has its own issues; this will be discussed in this part of the chapter. The game design has many aspects to it especially the appearance of it. The approach that was taken will be discussed.

5.5.1 Initial Design

Before programming of the game started, it was essential to determine what the maze would look like. As with any project, planning should be enforced first. So a design was created using a simple drawing package. When learning the programming language, it was realised that the design can be more intricate, as the language allowed the use of complexity. Figure 5.4 shows the early stage of designing the maze. It is in birds eye view, but one of the aims of this project was to create the game in first person view. It was designed like this so that the whole maze could be seen, and easily created. Learning Java 3D by examples gave an insight of what can be achieved, so another design of the maze was created but this time on paper and a little more complicated. This design is shown in Appendix C.
Figure 5.4: A simple maze design.

5.5.2 Wall Design

One of the requirements in creating such a game was to use primitive shapes. Primitive shapes are basic shapes that are defined in the Java library. These shapes are boxes, cones, cylinders, spheres and even text 2D. There is an alternative method by using a map. The map is a design that is created as a GIF image and uses characters that interprets it as walls. This method is a little less complicated and confusing as the map is created before the game is executed rather than using primitive shapes which is created as the game is executed simultaneously.

The design of the walls is constructed by a primitive shape, a box. The box size is defined by the right hand system discussed earlier in this chapter. Placing the box at a certain place in the scene was the trickiest test so far. Each wall had to be placed at an exact position according to the design of the maze in Appendix C. If the co-ordinates of a wall was slightly out then the consistency of the design would look terrible and shorten the time a player would play the game.

The co-ordinates were written down on paper along with the design of the maze, so when it came to placing a wall it wasn’t too much of a predicament. Looking at figures 5.5 and 5.6 it can be seen what a difference of 1 unit can make to the placing of a wall in the game.
Figure 5.5: Wrong co-ordinate placing  Figure 5.6: Right co-ordinate placing

The wall in the middle of diagram 5.5 is a little out; another meaning is that the wall is not in line with the rest of the walls. The wall in figure 5.6 was adjusted so that it was in line with the rest of the walls. This was a problem when it came to programming, but it was solved by looking at each wall while the program is executed and double checked to see if it should be there or not.

Once the wall design was implemented from paper to computer, the next stage was to add colour to the walls. Various colours were tested out varying from red, blue, yellow to orange. The colour that was chosen was green. The concept of the game was to keep it simple, which would result in less loading time. Adding the colour green made it visually appealing and also makes it easy on the eye. Adding the colour green made the affect of a 3D game appear 2D, so lighting had to be applied in the scene to make the walls appear 3D. Below is a sample code of defining colour taken from the source code of the project:

```java
Color3f ambientColour = new Color3f(0.5f,1.0f,0.1f);
Color3f emissiveColour = new Color3f(0.0f,0.0f,0.0f);
Color3f specularColour = new Color3f(0.0f,0.0f,0.0f);
Color3f diffuseColour = new Color3f(0.2f,0.0f,0.2f);
float shininess = 0.0f;
```

5.5.3 Lighting

To view the walls and to give it a more realistic appearance, lighting must be applied. Figures 5.5 and 5.6 used a simple light to see the walls. The above code is also said to use light to
emerge the colour onto a surface. The way an appearance is shown is based on two components: the surface properties and the light itself.

Surface light is used in the code above. There are four main types of surface light that has been applied.

- Ambient colour: This is the colour that affects all areas of a surface equally.
- Diffuse colour: This is like a reflection; it is used to reflect light evenly on a surface. It can be a completely odd colour, but to make it realistic it should be a similar colour to the surface colour of the wall. In this case it is a lighter shade of the green.
- Specular colour: This also can be known as shininess. It is used to produce highlights of a surface colour, like polished snooker balls.
- Emissive colour: This defines the amount of light produced by the surface.

The other type of lighting that is used in the game is the light itself in the scene. Looking at the code below, this is used to produce two types of light in the game: AmbientLight and DirectionalLight.

```java
Color3f ambLightColour = new Color3f(0.5f, 0.5f, 0.5f);
AmbientLight ambLight = new AmbientLight(ambLightColour);
ambLight.setInfluencingBounds(bounds);
Color3f dirLightColour = new Color3f(1.0f, 10.0f, 1.0f);
Vector3f dirLightDir = new Vector3f(0.0f, -1.0f, -1.0f);
DirectionalLight dirLight = new DirectionalLight(dirLightColour, dirLightDir);
dirLight.setInfluencingBounds(bounds);
b.addChild(ambLight);
b.addChild(dirLight);
```

- Ambient light: This lights up the whole scene (game) equally, of a specified colour.
- Directional light: This is light that is used from a distance away, such as the light from the sun.

Adding lighting created a problem because using different light sources; they can emerge together and create a colour that was not intended. For example the sun in the game had an ambient colour set to a shade of yellow, but when directional light was added to the game the colour of the sun changed to very pale yellow, which almost looked like a creamy colour. Working to get the best colour for the sun took large amount of time. A trial and error process
was used, and eventually the colour of the sun was looking as it should be. Combining all the lighting affects generates the game to be more of a realistic view.

5.5.4 Textures

The use of textures in a game can have immersed affects to the experience of gaming. We can shape a sphere to make it look like the planet Earth from a distance, by just using a 2D map of the Earth. This saves a great deal of rendering time, for example to texture an image of a pavement by just using colours and light sources it would take large amount of coding to achieve this. To texture a picture of a pavement on the shape would reduce the amount of coding used.

The texture used in the game was of the ground the maze lays on. It was textured as a pavement. Adding more textures to the game increased the loading time. As this game is intended to be played over the Internet, adding lots of textures would increase the downloading time and make the user impatient and would steer the user away to surf for something else. Below is the code that was used to point to a file for texture.

```java
try {Texture tex = new TextureLoader("floor.jpg", this).getTexture();
    app32.setTexture(tex);
}

```

Box Ground = new Box (100f,0.01f,100f,Primitive.GENERATE_TEXTURE_COORDS | Primitive.GENERATE_NORMALS_INWARD,app32);

The idea of having textured walls was initially a great one, but texturing primitive shapes was a tricky method. Texturing a primitive shape only textured one side of a box in the game. A 2D image of a brick wall in a .JPG format was tested onto a wall of the game. As the box has 6 sides only one side had the image of the brick wall. This was tried with different ways of coding, but even though it still did not have the effect that was intended. It was decided that only light would be used to add colour to the walls, as it was the only way forward.
5.5.5 Movement

Java 3D is to do with graphics. Normally the graphic is still and is viewed just for its admiration, but what if the requirement is to move around the scene. This is required to make the experience of a maze game more realistic.

Adding navigation to the scene to make it look like a first person view was a task that was easily accomplished. Only a few lines of code were required and it is written below.

```
BoundingSphere movingBounds = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
BoundingLeaf boundLeaf = new BoundingLeaf(movingBounds);
...
...
...
...
KeyNavigatorBehavior keyNav = new KeyNavigatorBehavior(viewXfmGroup);
keyNav.setSchedulingBounds(movingBounds);
viewBranch.addChild(keyNav);
```

Adding movement in the game, it allowed the player to walk through walls. So what needed to be done was to add a behaviour to the game. The behaviour that should be used in this instance was called collision detection. Collision detection was researched into depth, to find a code and to implement it into the game. The way it was carried out was to add a separate class and it would be called when it was needed from the main Java file. Step by step it was looking like this would work. Adding the behaviour was a success, when a player walked through the wall the class collision detection realised this was happening. It was programmed so that in the Java console when a player walks through the wall it would say ‘a collision has been detected’.

The next step was to implement this behaviour so that the player does not walk through the wall. This was a setback, as this was not achieved. Numerous coding was tested, and even help was asked through Java forums, but to no avail. It was decided that this task would be looked back into at a later stage, to try and get it to work.

5.5.6 Sound

Adding sound to a graphical scene may seem strange. It does not sound strange when there is a game in production. Adding sound creates the atmosphere the graphical side cannot do. Sound makes the game more fascinating and in cases it can play part of the suspense.
In the project sound was added as a background sound. The object of the game is to get from start to finish, by investigating the right routes and avoiding dead-ends. Therefore it needed a background sound that involved this instance. The original Pink Panther theme was used to create this atmosphere.

Adding this background sound was an achievement, the next step was to loop the track as it only lasted thirty seconds. Lots attempts were tried and tested, but the coding that was implemented in the game was correct. The reason why it did not work was beyond the authors control, as there was a bug in Java 3D to this aspect of sound. So what was thought of was to have a lengthier track. The track that was used was still Pink Panther, but now it lasted for a minute and a half.

5.5.7 Text 2D

Adding text to a scene instructs a user of what to do. It may seem bizarre to add text but any game available has some sort of text one way or another.

The text that was used in the game was a form of 2D text. 2D text was preferred over 3D text because it seemed appropriate as it told the user this is the starting point and this is the finish point. Figure 5.7 on the next page illustrates this.

![Figure 5.7: The starting post using text 2D](image)

Adding this text created a problem, from distance the text would distort. This may be the case of having not a powerful PC. There was not much the author could have done to fix this, so as the text was still readable it was decided that this was not a problem that affected the game playing. Therefore it was decided that it would be looked at in future development.
5.5.8 Background

Adding a background adds interest to the game by showing definition to the scene. Firstly a background image of the sky was inserted but this would only show on one side of the scene. When the player turns around it would just be a black scene. It was then thought to put the texture in a shape of a sphere and wrap it around the maze. But when this coding was complete it took a very long time to load. So it was decided to add a background colour rather than a textured image to decrease loading time.

5.5.9 Final Game Product

With all the designing and creation of the game complete, a proud moment was released. Figure 5.8 shows the final game creation. The walls are of two colours to distinguish 3D affect; a sun was created to give the game an outdoor feel, with different types of lighting involvement. The floor is textured from a file and also something that cannot be noticed is the sound, when the game is executed it would play. Screenshots of the game can be found in Appendix E.

![Final game created](image)

Figure 5.8: Final game created

5.6 Java 3D in a Browser

Most browsers today can accompany Java in a browser, by a form of an applet. Browsers such as Internet Explorer have a Java Virtual Machine (JVM) shipped with it. JVM is a technology that enables Java applications to run on cross based platforms. This allows the player efficient playing of the game without needing to install necessary files.
The way it works is that a webpage is created, and in that webpage is the applet placed accordingly. Adding an applet is a simple task, which is already a function in Dreamweaver 4. When the webpage is put on a server, it requires the class file of the application to be destined in the same directory and folder. When a user surfs the web and comes across the webpage the applet is downloaded and a space in the browser is made for it to be displayed. Figure 5.9 shows how the JVM works.

![Figure 5.9: How JVM works](image)

The game is created both as an applet and as an application. A Java application is a standalone program that runs without the need for a browser.

### 5.7 Webpage Design

When creating the webpage it was fundamental that the layout was perfect. Important factors of a webpage are colour schemes, graphics, fonts and its sizes even consistency. With all these in mind a successful webpage can be constructed, and have a massive impact with users viewing it.

#### 5.7.1 Layout

A layout was first designed on paper to be implemented on to the computer. When this was achieved specific components of the webpage were placed using Dreamweaver. The function to add layers was used to achieve this.
5.7.2 Colour Scheme

The colour scheme was important and that it matched the colour of the game scene. Adding bright colours made it hurtful to the eye and reading text was a struggle. Many colours were tested but the best one was blue as it matches the game colour or the colour of the scene. The text that was used in the website was a contrast to blue as this could be viewed and read easily.

5.7.3 Plug-Ins

When the webpage is first viewed and it’s the first time the computer has seen Java, it will automatically prompt the user to install the required files (plug-ins) for the game to work. It is a quick and simple process that only requires a few clicks of the mouse. Figure 5.9 shows this process.

5.7.4 Title

The title reflected the game itself. Many fonts were used but it did not have the look that the author was intended for, which was to make it look simple and fun. So with many attempts to find a suitable font, it was certain that the Internet should be looked at. From previous projects the author has worked on, it was noted that FlamingText.com was a useful resource to create a fitting logo. The logo is illustrated in figure 5.10.

Figure 5.10: Layout of webpage
5.8 Summary

This chapter has looked at how the game was created using different components and elements. All the elements that are used to create a game such as sound play a crucial role into building the game. With these entire elements incorporated in one the player can experience a fascinating game. The webpage was discussed and defined how the game would blend in with the webpage to give it the atmosphere that is required to challenge a player into playing more of the game.
6 TESTING

6.1 Introduction

With any application once it is completed it needs to go through numerous stages of testing. There are many different ways of testing an application from testing speeds, loading times, sounds to just the look of it. With this project it was felt that as it was created both as an application and as a web page that more tests should be conducted. The two types of testing that were involved in this project were functional testing and structural testing.

6.2 Functional Testing

This is also known as black box testing. This type of testing doesn’t test the internal structure of the application. Black box testing should not be performed by the author of the program who knows too much about the program internals. Therefore it was approached to people that were not involved with any gaming experience. The reason why this was carried out was because as they have little experience with games, they could give a true and unbiased response whether they liked it or not. A questionnaire was conducted and passed on to the people that played with the game. The questionnaire and results are in Appendix D.

The advantages of this type of testing include:

- The creator of the application is independent from the tester, so it would give an unbiased view.
- The tester does not need to know any programming techniques.

The disadvantages of this type of testing include:

- If a certain test case has already been reviewed by the programmer then it is made redundant if it is tested again.
- Creating a test case after the tester has worked with the program can be difficult.

6.3 Structural Testing

This is also known as white box testing. This is specifically to do with software testing. It’s a technique whereby explicit knowledge of the internal workings of the project is being tested. Unlike functional testing, structural testing uses specific knowledge of programming codes to
examine outputs. The test is accurate only if the programmer knows what the program is supposed to do.

Structural testing was used in this project but was not documented. The way it was tested was when programming the game, when an error was surfaced it was looked at and then rectified. For example adding a background colour to the game gave its fair share of compiling errors, these were simple errors such as forgetting to add a comma in a line of a code. What was achieved was to look at the code and to solve this at that certain time. With programming especially programming with Java, every line of code has to be case sensitive and each character has got its own element to it. Creating and revising new coding is not permitted. Everything has got to be the way it should be.

6.4 Testing a Game

When testing applications it is too broad of an area to judge, so a view on how to achieve a game was looked at especially for gaming in general. Factors that are tested are pointed below:

- The graphical, rendering, animation aspect.
- Sound involvement.
- Intelligence of the game.
- Game controls.
- The front end of the game (looks).

With all these in mind testing could be accomplished for games. Any errors that are emerging can be solved and re-tested to ensure the aim of this project is met.

6.5 Questionnaire

As mentioned earlier in this chapter a questionnaire was devised, to get feedback. With feedback the project could be evaluated, and any negative points raised could be made into positive points. The questionnaire was passed out to friends and family of the author, some of the questions that were asked were:

- Does the game look interesting?
- Does it make you want to play again?
- Is any colour in the game distracting?
- Does the webpage look motivating?
The analysis from receiving feedback from questionnaires was a success. It shows the aims and objects of this project were met and the results can be found in Appendix D.

6.6 Summary

Testing has proven to be a huge improvement to creating a project such as this. It has given the author valuable insight of how to go from a negative point to a positive point. Using the Waterfall methodology, which is explained in chapter 4, it has proven to be of a success. Testing while producing this project wasn’t a must but it came naturally to the author. The following chapter looks at testing in a little bit more detail, such as platform testing, which is running the game on different specification computers.
7 CONCLUSION

7.1 Introduction

This chapter is a review of the complete project. It will discuss the aims and objectives which were analysed in chapter 1.2 and 1.4. The author will discuss the accomplishments of the design and implementation to the project, as well as project planning. There will be an outline of the alternative methods that could have been approached and future work that the author could strive on.

7.2 Project Outline

The project was considered a success as the aims and objectives that were highlighted in chapter one were completed to the author’s satisfaction.

The aim of this project was to create a computer maze game using Java 3D. The research that was carried out was beneficial to the project as the author was educated with the programming language Java 3D and the potential it can have on any applications. Once the game was completed a player can be made aware of the advantages of Java 3D, and what it can accomplish. Using various aspects of Java 3D such as lighting and texturing, the game created an appealing curiosity to it, which will draw people of all ages to play it.

The full working source code of the game was completed which is made available in Appendix B, with comments to help a novice user of Java 3D understand it. A full working webpage was created with a certain type of design creativity implemented. The game has an inquisitiveness to it, which makes a player want to have a go at the game. The project can be made of an ongoing project as all the coding is available and the game can have a next degree of intelligence to it.

The game can be available to any player or an Internet surfer once it is put on a server. The required specification of a computer to play the game is to a minimum and as Java has the ability to be used on any platform with the help of JVM, nobody would feel left out. Even if the player’s computer hasn’t seen Java before, an automatic prompt will pop-up and make the user aware that certain files needs to be installed to the machine in order to access the game. The only aspect the player needs to be made aware of is that the machine needs to have a graphics card of at least 32MB and has to have 3D capabilities, such as Direct X.
A design methodology was used in this project to help the smoothness of running of it. The Waterfall methodology was implemented in every aspect of the project, which identified certain aspects that needed to be accomplished, before another task could get underway. Another advantage to using this methodology was the ability to plan, therefore the project stayed on course.

### 7.3 Analysis of Project

The aim and objectives were met in this project, but certain tasks had to be critically reviewed. One of the major factors that had a drawback to creating and playing a game in Java 3D, was that it needs to have a graphics card of at least 32MB on the computer. Now we all know that not everyone has this type of specification. But as technology is improving in the digital world it is highly unlikely that Internet surfers would have such a low specification on their machine. Even though with this in mind it was beneficial for the author to test the game on various types of computers that could be able to play the game.

In figure 7.1, it shows the various specifications of computers that the game was tested on. Now all the computers did have a graphics card of at least 32MB and have the operating system Windows.

<table>
<thead>
<tr>
<th>Processor</th>
<th>RAM</th>
<th>Operating system</th>
<th>Graphics card</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4GHz AMD Athlon</td>
<td>768MB</td>
<td>Windows XP</td>
<td>64 MB</td>
<td>Works as expected</td>
</tr>
<tr>
<td>2.4Ghz Intel Pentium</td>
<td>512MB</td>
<td>Windows XP</td>
<td>128 MB</td>
<td>Smoother running</td>
</tr>
<tr>
<td>1.7GHz Intel Celeron</td>
<td>128MB</td>
<td>Windows ME</td>
<td>32 MB</td>
<td>Slower running</td>
</tr>
<tr>
<td>2.8 Intel Pentium</td>
<td>512MB</td>
<td>Windows 2000</td>
<td>256 MB</td>
<td>Excellent running</td>
</tr>
<tr>
<td>1.0GHz Intel Pentium</td>
<td>128 MB</td>
<td>Windows 98</td>
<td>8MB</td>
<td>Did no run at all</td>
</tr>
</tbody>
</table>

**Figure 7.1:** Specification of computers tested with 3D maze game.

From the table it can be seen that the game runs on any specification as long it has a minimum graphics card of 32MB, unfortunately the last computer tested, the game did not run it just
showed a blank screen. The on board processor of the graphics card does the calculations of 3D modelling.

Creating the game source code was one of the greatest achievements the author has experienced with computers. It gave the opportunity to solve compiling errors that appeared when least expected. One of the greatest achievements in this project was the ability to form an original idea of a 3D maze in first person view.

All functionality testing was completed without a hitch. Checking to see if the game had the right colours to it, or verifying if the game had a certain enjoyment to it was tested. One of the things tested was to see if the game could view objects far away, but this did not pass. When the game is first executed the player only sees what the coding is programmed to do which is shown in figures 7.2 and 7.3. The first diagram shows that the back wall is not viewed as well as the sun, but as soon as the player walks forward the wall and sun appear.

![Figure 7.2: Game with no back wall.](image1)

![Figure 7.3: Game with back wall.](image2)

What needs to be looked at was something called Viewpoint. Unfortunately due to the fact that information on this subject was vague, the author did not have time to experiment with Viewpoint. But this is noted down as a future development.

When lines of coding were not compiling, alternative methods was intervened. Such as adding background colour. Originally it was thought of having an image, but as it got complicated with coding and efficiently in loading time the built in method of background colour in Java was initiated.

One downside to adding sound was that it would not loop; this was a factor beyond the author’s control. When researching on this instance it was found that there was a bug in Java 3D with
Java Sound Mixer. This bug could not be solved by the author. To view this bug it can be reached on at this URL: http://java.sun.com/products/java-media/3D/java3d-bugs.html.

7.4 Project Planning

Even though no project goes to 100% as planned, as little things can cause disruption, this project is believed to be a success due to effective planning and control. All tasks were planned with extra time in mind to complete them. The Gantt chart in Appendix A is the final version that was created out of 2. It shows that tasks were allocated a time to complete and that control was put into motion.

Regular meetings with the project supervisor, was a gain to the project. This focused the author to stay in the right direction and path to complete the aims and objectives defined in chapter one.

7.5 Alternative Methods

When constructing a game, ideas flow into the mind. These ideas could be tasks that could be done a different way to increase popularity and achievement. These tasks could be programmed to work as an alternative method, not necessary to make the project easier but to have the same level of success.

One of the alternative methods that could have been approached was to create the game using a map based version to the game. With this it is meant that it is an alternative of using primitive shapes. The reason why primitive shapes were used was because it was an aim set by the supervisor, to test whether it could be accomplished by the author.

Another alternative process could be to use another language such as C++. Learning a new language can have tremendous benefits to the author as it could bring new opportunities in the working world and have the ability to create new projects.

Instead of using Java Sound Mixer another way could have been implemented. The reason for this was so that sound would be able to loop. Another way is add 3D sound which is a library of Java 3D.
7.6 Further Development

As time is a major factor in any project, as it has deadlines involved, certain characteristics can only be accomplished. But with the possibility of extra time and involvement greater achievements can be achieved.

One factor to increasing time was to the ability to get collision detection in Java 3D in full working order. This was worked on but time was a disadvantage to the author, as it would not work as expected. Collision detection is a process that stimulates the program to know when an event occurs. In this case the process could have been used so that the user does not walk though walls in the maze as this does happen in the game. If this was developed then even more could come out of it, such as when the player walks to the wall, a sound could be heard such as a bang noise to indicate a collision has occurred.

As mentioned above, Viewpoint could have been researched into more depth if time allowed it. This would give the game a full 3D environment feeling to any player. It would also look more detailed in the game scene.

The game was tested on different specification computers, but not tested with machines that have different operating systems such as Linux and Macintosh. With a bit of funds this could be tested by third parties that have the resources and equipment available.

A final future development could be to create a full working game that could be sold in game stores to generate profit. This would involve the author and a dedicated team to work full time on such a project, to compete with other game manufacturers.

7.7 Summary

This project has proven to be an accomplishment to the author by being able to meet aims and objectives planned before the project got underway. The invaluable experience that has been taught was the way of effective planning, problem solving and learning a new language. With this experienced gained it could be bought on to future developments with other similar tasks. Overall the entire project has brought satisfaction to the author for all the work completed in fulfilment.
8 REFERENCES

History of Mazes:
http://www.btinternet.com/~mazes/history.htm

The History of Mazes:
http://www.unmuseum.org/maze.htm

Human Computer Interaction:
http://sigchi.org/cdg/cdg2.html

Methodologies:
http://www.iqx.com/methodologies.htm

The Java 3D API Specification:
http://java.sun.com/products/java-media/3D/forDevelopers/J3D_1_3_API/j3dguide/index.html

Project Planning:
http://oldeee.see.ed.ac.uk/~gerard/Management/art8.html

Right Hand System:
http://www.vard.org/mono/gait/sout-f04.jpg

Design Methodologies:

3D Maze Game:
http://dev-javaboutique.internet.com/3DMaze/

3D Maze Game:
http://amadeus.uprm.edu/~goopm/MainProg.html

The Java Community:
http://www.j3d.org/

Java Gaming Forum:
http://www.javagaming.org/cgi-bin/JGNetForums/YaBB.cgi?board=3D

Java Sun Forum:
http://forum.java.sun.com/forum.jsp?forum=21

Introduction to Programming with Java 3D:
http://java.sun.com/products/java-media/3D/collateral/class_notes/slides/mt0000.htm
9 BIBLIOGRAPHY


### Appendix A: Gantt chart Part 1

| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| **Registration of FYP** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **Project Title Allocation** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

#### FEASIBILITY STUDY
- Research into hardware required
- Research into software required
- Submit Feasibility Report
- Complete Project specification

#### SYSTEM DESIGN
- Create a box to represent a wall
- Learn how to use co-ordinates in Java 3D
- Plot co-ordinates to represent walls
- Add texture to walls
- Add background colour
- Add background sound
- Add lighting
- Create movement in the game
- Apply collision behaviour
- Create a webpage
- Add the game to the webpage

---

**Deadlines**

**Tasks**

**Exam Period**

**Holidays**
### Continued Gantt chart Part 2

| Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
|------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| PRESENTATION SEMINAR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Prepare for project seminar | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Seminar | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TESTING | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Test game for errors | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Correct errors | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Re-test | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PROJECT DEMONSTRATION | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Prepare for demonstration | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Demonstration | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PROJECT REPORT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Hand in report | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| POSTER SESSIONS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Prepare for poster session | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Poster session | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| MAINTENANCE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Thought-out Project**

- **Deadlines**
- **Tasks**
- **Exam Period**
- **Holidays**
APPENDIX B
Appendix B – Full Source Code

/* This imports the Java3D classes */
import javax.media.j3d.*;
/* Import the vector classes so we can perform simple maths */
import javax.vecmath.*;
/* Import GraphicsConfiguration so that the hardware can work
according to the coding */
import java.awt.GraphicsConfiguration;
/* Imports the utility universe from SUN */
import com.sun.j3d.utils.universe.*;
/* Import the AWT classes so we can create window etc. */
import java.awt.*;
/* Import the AWT events so we can use a button to exit the program */
import java.awt.event.*;
/* Import the behavior class so we move around the game */
import com.sun.j3d.utils.behaviors.keyboard.*;
/* Import the class geometry so we can create primitive shapes */
import com.sun.j3d.utils.geometry.*;
/* Import the java sound class */
import com.sun.j3d.audioengines.javasound.*;
/* Import the java texture class */
import com.sun.j3d.utils.image.TextureLoader;

public class Maze3D extends Frame implements ActionListener{

GraphicsConfiguration config =
SimpleUniverse.getPreferredConfiguration();
/* Canvas3D is used to display the scene */
Canvas3D myCanvas3D = new Canvas3D(config);
/* This adds a button to quit the game */
Button exitButton = new Button("Exit");
/* Creates the bounds for the scene */
BoundingSphere bounds = new BoundingSphere(new Point3d(10,0,0), 1000);

private BranchGroup buildViewBranch(Canvas3D c) {
/* Create a BranchGroup. A BranchGroup is a node in
a Tree data structure that can have child nodes */
BranchGroup viewBranch = new BranchGroup();
Transform3D viewXfm = new Transform3D();
/* This starts the player at coordinates 0,0,0 in the scene graph */
viewXfm.set(new Vector3f(0f,0f,0f));
/* This sets the transform group to be a parent */
TransformGroup viewXfmGroup = new TransformGroup(viewXfm);
/* This allows the capability of writing and reading the scene graph
so for example we can add the keyboard navigation in */
viewXfmGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
viewXfmGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
BoundingSphere movingBounds = new BoundingSphere(new Point3d(0,0,0,0), 100.0);
BoundingLeaf boundLeaf = new BoundingLeaf(movingBounds);
ViewPlatform myViewPlatform = new ViewPlatform();
viewXfmGroup.addChild(boundLeaf);
/* The phisical elements are created */
PhysicalBody myBody = new PhysicalBody();
PhysicalEnvironment myEnvironment = new PhysicalEnvironment();
/* This puts the scene together */
viewXfmGroup.addChild(myViewPlatform);
viewBranch.addChild(viewXfmGroup);
View myView = new View();
myView.addCanvas3D(c);
myView.attachViewPlatform(myViewPlatform);
myView.setPhysicalBody(myBody);
myView.setPhysicalEnvironment(myEnvironment);
/* A key navigation utility object is created and associated with
the view transform so that the it creates that first person movement */
KeyNavigatorBehavior keyNav = new KeyNavigatorBehavior(viewXfmGroup);
keyNav.setSchedulingBounds(movingBounds);
viewBranch.addChild(keyNav);
/* This creates the background colour */
Background back = new Background();
back.setColor(new Color3f(0.2f, 0.6f, 1f));
back.setApplicationBounds(bounds);
viewBranch.addChild(back);
/* Create a sounds mixer to use our sounds with and initialise it */
JavaSoundMixer myMixer = new JavaSoundMixer(myEnvironment);
myMixer.initialize();
return viewBranch;

/* This creates a bounds for the lights in the scene graph to the
parameter b */
This sets up the global lights */
private void addLights(BranchGroup b) {
/* Creates an ambient light. */
Color3f amLightColour = new Color3f(0.5f, 0.5f, 0.5f);
AmbientLight amLight = new AmbientLight(amLightColour);
amLight.setInfluencingBounds(bounds);
/* This sets the direction of the light */
Color3f dirLightColour = new Color3f(1.0f, 10.0f, 1.0f);
Vector3f dirLightDir = new Vector3f(0.0f, -1.0f, -1.0f);
DirectionalLight dirLight = new DirectionalLight(dirLightColour,
dirLightDir);
dirLight.setInfluencingBounds(bounds);
/* This adds the light to the BranchGroup */
b.addChild(amLight);
b.addChild(dirLight);
}
/* This adds a continuous background sound to the branch group
Parameter b in the BranchGroup adds the sound to.
The parameter soundFile String is the name of the sound file */
private void addBackgroundSound (BranchGroup b,String soundFile) {
/* This create a media container to load the file */
MediaContainer musicContainer = new MediaContainer(soundFile);
/* This creates the background sound from the media container */
BackgroundSound music = new BackgroundSound(musicContainer,1.0f);
music.setSchedulingBounds(bounds);
music.setEnable(true);
/* Make the sound file loop */
music.setLoop(BackgroundSound.INFINITE_LOOPS);
/* Adds the parameter to the group music */
b.addChild(music);
}
private BranchGroup buildContentBranch() {
/* create a parent BranchGroup node for the walls, the sun, the ground etc. */
}
BranchGroup contentBranch = new BranchGroup();

/* Create an Appearance for each wall as well as the sun and the ground.
The Appearance object controls various rendering options such as the colour */

Appearance app1 = new Appearance();
Appearance app2 = new Appearance();
Appearance app3 = new Appearance();
Appearance app4 = new Appearance();
Appearance app5 = new Appearance();
Appearance app6 = new Appearance();
Appearance app7 = new Appearance();
Appearance app8 = new Appearance();
Appearance app9 = new Appearance();
Appearance app10 = new Appearance();
Appearance app11 = new Appearance();
Appearance app12 = new Appearance();
Appearance app13 = new Appearance();
Appearance app14 = new Appearance();
Appearance app15 = new Appearance();
Appearance app16 = new Appearance();
Appearance app17 = new Appearance();
Appearance app18 = new Appearance();
Appearance app19 = new Appearance();
Appearance app20 = new Appearance();
Appearance app21 = new Appearance();
Appearance app22 = new Appearance();
Appearance app23 = new Appearance();
Appearance app24 = new Appearance();
Appearance app25 = new Appearance();
Appearance app26 = new Appearance();
Appearance app27 = new Appearance();
Appearance app28 = new Appearance();
Appearance app29 = new Appearance();
Appearance app30 = new Appearance();
Appearance app31 = new Appearance();
Appearance app32 = new Appearance();
Appearance app33 = new Appearance();
Appearance app34 = new Appearance();
Appearance app35 = new Appearance();
Appearance app36 = new Appearance();
Appearance app37 = new Appearance();
Appearance app38 = new Appearance();
Appearance app39 = new Appearance();
Appearance app40 = new Appearance();

/* We need to define colours for various items in the scene such as the walls */

/* Wall colour */
Color3f ambientColour = new Color3f(0.5f,1.0f,0.1f);
Color3f emissiveColour = new Color3f(0.0f,0.0f,0.0f);
Color3f specularColour = new Color3f(0.0f,0.0f,0.0f);
Color3f diffuseColour = new Color3f(0.2f,0.0f,0.2f);
float shininess = 0.0f;

/* Sun colour */
Color3f ambientColour2 = new Color3f(1.8f,4.0f,1.0f);
Color3f emissiveColour2 = new Color3f(0.0f,0.0f,0.0f);
Color3f specularColour2 = new Color3f(1.0f,2.0f,2.0f);
Color3f diffuseColour2 = new Color3f(3.0f, 2.5f, 2.5f);
float shininess2 = 500.0f;

/* Sign colour */
Color3f ambientColour3 = new Color3f(1.8f, 1.4f, 0.7f);
Color3f emissiveColour3 = new Color3f(0.0f, 0.0f, 0.0f);
Color3f specularColour3 = new Color3f(0.01f, 0.01f, 0.01f);
Color3f diffuseColour3 = new Color3f(0.0f, 0.0f, 0.0f);
float shininess3 = 0.0f;

/* Ground texture */
try {Texture tex = new TextureLoader("floor.jpg", this).getTexture();
    app32.setTexture(tex);
} catch (Exception e) {e.printStackTrace();
}

/* The Appearance needs to applied to each items in the scene */
appl1.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl2.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl3.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl4.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl5.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl6.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl7.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl8.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl9.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl10.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl11.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl12.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl13.setMaterial(new Material(ambientColour, emissiveColour,
diffuseColour, specularColour, shininess));
appl14.setMaterial(new Material(ambientColour, emissiveColour,

diffuseColour, specularColour, shininess));

app15.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app16.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app17.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app18.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app19.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app20.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app21.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app22.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app23.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app24.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app25.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app26.setMaterial(new Material(ambientColour, emissiveColour, diffuseColour, specularColour, shininess));
app27.setMaterial(new Material(ambientColour2, emissiveColour2, diffuseColour2, specularColour2, shininess2));
app28.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app29.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app30.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app31.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app32.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app33.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app34.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app35.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app36.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app37.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app38.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app39.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app40.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app41.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app42.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app43.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app44.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app45.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app46.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app47.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app48.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app49.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app50.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app51.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app52.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app53.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app54.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app55.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app56.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app57.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app58.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app59.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app60.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app61.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app62.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app63.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app64.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app65.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app66.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app67.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app68.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app69.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app70.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app71.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app72.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app73.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app74.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app75.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app76.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app77.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app78.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app79.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app80.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app81.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app82.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app83.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app84.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app85.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app86.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app87.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app88.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app89.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app90.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app91.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app92.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app93.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app94.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app95.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app96.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app97.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app98.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app99.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
app100.setMaterial(new Material(ambientColour3, emissiveColour3, diffuseColour3, specularColour3, shininess3));
diffuseColour3,specularColour3,shininess3));
app39.setMaterial(new Material(ambientColour3,emissiveColour3,
    diffuseColour3,specularColour3,shininess3));

/* Each item needs to be of a size, the box appearance for example needs three values for X,Y and Z to create a wall */
Box Wall1 = new Box(1f,1f,30f,app1);
Box Wall2 = new Box(29f,1f,1f,app2);
Box Wall3 = new Box(1f,1f,29f,app3);
Box Wall4 = new Box(28f,1f,1f,app4);
Box Wall5 = new Box(26f,1f,1f,app5);
Box Wall6 = new Box(1f,1f,28f,app6);
Box Wall7 = new Box(3f,1f,1f,app7);
Box Wall8 = new Box(20f,1f,1f,app8);
Box Wall9 = new Box(1f,1f,8f,app9);
Box Wall10 = new Box(1f,1f,14f,app10);
Box Wall11 = new Box(1f,1f,4f,app11);
Box Wall12 = new Box(1f,1f,1f,app12);
Box Wall13 = new Box(1f,1f,1f,app13);
Box Wall14 = new Box(1f,1f,1f,app14);
Box Wall15 = new Box(1f,1f,12f,app15);
Box Wall16 = new Box(1f,1f,1f,app16);
Box Wall17 = new Box(1f,1f,4f,app17);
Box Wall18 = new Box(1f,1f,1f,app18);
Box Wall19 = new Box(23f,1f,1f,app19);
Box Wall20 = new Box(1f,1f,1f,app20);
Box Wall21 = new Box(2f,1f,1f,app21);
Box Wall22 = new Box(2f,1f,1f,app22);
Box Wall23 = new Box(1f,1f,5f,app23);
Box Wall24 = new Box(2f,1f,1f,app24);
Box Wall25 = new Box(1f,1f,5f,app25);
Box Wall26 = new Box(2f,1f,1f,app26);
Box Wall27 = new Box(1f,1f,5f,app27);
Box Wall28 = new Box(1f,1f,5f,app28);
Box Wall29 = new Box(2f,1f,1f,app29);
Box Wall30 = new Box(28f,1f,1f,app30);
Box Wall31 = new Box(1f,1f,2f,app31);
Box Ground = new Box(100f,0.01f,100f,Primitive.GENERATE_TEXTURE_COORDS |
    Primitive.GENERATE_NORMALS_INWARD,app32);
Sphere Sun = new Sphere(2f,app33);
Text2D Start = new Text2D("Start",new Color3f(0.0f, 0.0f,
    0.0f),"Arial", 50, Font.BOLD);
Text2D Finish = new Text2D("Finish",new Color3f(0.0f, 0.0f,
    0.0f),"Arial", 50, Font.BOLD);
Box Sign1a = new Box(0.4f,0.2f,0.01f,app36);
Box Sign1b = new Box(0.09f,0.5f,0.05f,app37);
Box Sign2a = new Box(0.4f,0.2f,0.01f,app38);
Box Sign2b = new Box(0.09f,0.5f,0.05f,app39);

/* Each item needs to placed in the scene using the X,Y,Z geometry
   Basically it puts it all together */
Transform3D Group1Xfm = new Transform3D();
Group1Xfm.set(new Vector3d(-2,0,-40));
TransformGroup Group1 = new TransformGroup(Group1Xfm);

Transform3D Group2Xfm = new Transform3D();
Group2Xfm.set(new Vector3d(32,0,-11));
TransformGroup Group2 = new TransformGroup(Group2Xfm);
Transform3D Group3Xfm = new Transform3D();
Group3Xfm.set(new Vector3d(60, 0, -41));
TransformGroup Group3 = new TransformGroup(Group3Xfm);

Transform3D Group4Xfm = new Transform3D();
Group4Xfm.set(new Vector3d(27, 0, -69));
TransformGroup Group4 = new TransformGroup(Group4Xfm);

Transform3D Group5Xfm = new Transform3D();
Group5Xfm.set(new Vector3d(29, 0, -16));
TransformGroup Group5 = new TransformGroup(Group5Xfm);

Transform3D Group6Xfm = new Transform3D();
Group6Xfm.set(new Vector3d(54, 0, -29));
TransformGroup Group6 = new TransformGroup(Group6Xfm);

Transform3D Group7Xfm = new Transform3D();
Group7Xfm.set(new Vector3d(56, 0, -42));
TransformGroup Group7 = new TransformGroup(Group7Xfm);

Transform3D Group8Xfm = new Transform3D();
Group8Xfm.set(new Vector3d(29, 0, -21));
TransformGroup Group8 = new TransformGroup(Group8Xfm);

Transform3D Group9Xfm = new Transform3D();
Group9Xfm.set(new Vector3d(4, 0, -28));
TransformGroup Group9 = new TransformGroup(Group9Xfm);

Transform3D Group10Xfm = new Transform3D();
Group10Xfm.set(new Vector3d(10, 0, -36));
TransformGroup Group10 = new TransformGroup(Group10Xfm);

Transform3D Group11Xfm = new Transform3D();
Group11Xfm.set(new Vector3d(7, 0, -35));
TransformGroup Group11 = new TransformGroup(Group11Xfm);

Transform3D Group12Xfm = new Transform3D();
Group12Xfm.set(new Vector3d(31, 0, -26));
TransformGroup Group12 = new TransformGroup(Group12Xfm);

Transform3D Group13Xfm = new Transform3D();
Group13Xfm.set(new Vector3d(27, 0, -31));
TransformGroup Group13 = new TransformGroup(Group13Xfm);

Transform3D Group14Xfm = new Transform3D();
Group14Xfm.set(new Vector3d(31, 0, -36));
TransformGroup Group14 = new TransformGroup(Group14Xfm);

Transform3D Group15Xfm = new Transform3D();
Group15Xfm.set(new Vector3d(48, 0, -37));
TransformGroup Group15 = new TransformGroup(Group15Xfm);

Transform3D Group16Xfm = new Transform3D();
Group16Xfm.set(new Vector3d(51, 0, -47));
TransformGroup Group16 = new TransformGroup(Group16Xfm);

Transform3D Group17Xfm = new Transform3D();
Group17Xfm.set(new Vector3d(54, 0, -50));
TransformGroup Group17 = new TransformGroup(Group17Xfm);
Transform3D Group18Xfm = new Transform3D();
Group18Xfm.set(new Vector3d(57,0,-51));
TransformGroup Group18 = new TransformGroup(Group18Xfm);

Transform3D Group19Xfm = new Transform3D();
Group19Xfm.set(new Vector3d(32,0,-55));
TransformGroup Group19 = new TransformGroup(Group19Xfm);

Transform3D Group20Xfm = new Transform3D();
Group20Xfm.set(new Vector3d(4,0,-52));
TransformGroup Group20 = new TransformGroup(Group20Xfm);

Transform3D Group21Xfm = new Transform3D();
Group21Xfm.set(new Vector3d(7,0,-49));
TransformGroup Group21 = new TransformGroup(Group21Xfm);

Transform3D Group22Xfm = new Transform3D();
Group22Xfm.set(new Vector3d(13,0,-41));
TransformGroup Group22 = new TransformGroup(Group22Xfm);

Transform3D Group23Xfm = new Transform3D();
Group23Xfm.set(new Vector3d(16,0,-45));
TransformGroup Group23 = new TransformGroup(Group23Xfm);

Transform3D Group24Xfm = new Transform3D();
Group24Xfm.set(new Vector3d(19,0,-49));
TransformGroup Group24 = new TransformGroup(Group24Xfm);

Transform3D Group25Xfm = new Transform3D();
Group25Xfm.set(new Vector3d(22,0,-45));
TransformGroup Group25 = new TransformGroup(Group25Xfm);

Transform3D Group26Xfm = new Transform3D();
Group26Xfm.set(new Vector3d(25,0,-41));
TransformGroup Group26 = new TransformGroup(Group26Xfm);

Transform3D Group27Xfm = new Transform3D();
Group27Xfm.set(new Vector3d(28,0,-45));
TransformGroup Group27 = new TransformGroup(Group27Xfm);

Transform3D Group28Xfm = new Transform3D();
Group28Xfm.set(new Vector3d(36,0,-49));
TransformGroup Group28 = new TransformGroup(Group28Xfm);

Transform3D Group29Xfm = new Transform3D();
Group29Xfm.set(new Vector3d(34,0,-44));
TransformGroup Group29 = new TransformGroup(Group29Xfm);

Transform3D Group30Xfm = new Transform3D();
Group30Xfm.set(new Vector3d(29,0,-63));
TransformGroup Group30 = new TransformGroup(Group30Xfm);

Transform3D Group31Xfm = new Transform3D();
Group31Xfm.set(new Vector3d(50,0,-66));
TransformGroup Group31 = new TransformGroup(Group31Xfm);

Transform3D Group32Xfm = new Transform3D();
Group32Xfm.set(new Vector3d(0,-1,-30));
TransformGroup Group32 = new TransformGroup(Group32Xfm);

Transform3D Group33Xfm = new Transform3D();
Group33Xfm.set(new Vector3d(5,10,-70));
TransformGroup Group33 = new TransformGroup(Group33Xfm);

Transform3D Group34Xfm = new Transform3D();
Group34Xfm.set(new Vector3d(-1.25,-0.4,-8.93));
TransformGroup Group34 = new TransformGroup(Group34Xfm);

Transform3D Group35Xfm = new Transform3D();
Group35Xfm.set(new Vector3d(57.70,0.4,-69.93));
TransformGroup Group35 = new TransformGroup(Group35Xfm);

Transform3D Group36Xfm = new Transform3D();
Group36Xfm.set(new Vector3d(-1,-0.3,-8.94));
TransformGroup Group36 = new TransformGroup(Group36Xfm);

Transform3D Group37Xfm = new Transform3D();
Group37Xfm.set(new Vector3d(-1,-0.5,-9));
TransformGroup Group37 = new TransformGroup(Group37Xfm);

Transform3D Group38Xfm = new Transform3D();
Group38Xfm.set(new Vector3d(58,-0.3,-69.94));
TransformGroup Group38 = new TransformGroup(Group38Xfm);

Transform3D Group39Xfm = new Transform3D();
Group39Xfm.set(new Vector3d(58,-0.5,-70));
TransformGroup Group39 = new TransformGroup(Group39Xfm);

/* Applies each group to a certain child */
Group1.addChild(Wall1);
Group2.addChild(Wall2);
Group3.addChild(Wall3);
Group4.addChild(Wall4);
Group5.addChild(Wall5);
Group6.addChild(Wall6);
Group7.addChild(Wall7);
Group8.addChild(Wall8);
Group9.addChild(Wall9);
Group10.addChild(Wall10);
Group11.addChild(Wall11);
Group12.addChild(Wall12);
Group13.addChild(Wall13);
Group14.addChild(Wall14);
Group15.addChild(Wall15);
Group16.addChild(Wall16);
Group17.addChild(Wall17);
Group18.addChild(Wall18);
Group19.addChild(Wall19);
Group20.addChild(Wall20);
Group21.addChild(Wall21);
Group22.addChild(Wall22);
Group23.addChild(Wall23);
Group24.addChild(Wall24);
Group25.addChild(Wall25);
Group26.addChild(Wall26);
Group27.addChild(Wall27);
Group28.addChild(Wall28);
Group29.addChild(Wall29);
Group30.addChild(Wall30);
Group31.addChild(Wall31);
Group32.addChild(Ground);
Group33.addChildren(Sun);
Group34.addChildren(Start);
Group35.addChildren(Finish);
Group36.addChildren(Sign1a);
Group37.addChildren(Sign1b);
Group38.addChildren(Sign2a);
Group39.addChildren(Sign2b);

/* Adds each group to the contentBranch */
contentBranch.addChildren(Group1);
contentBranch.addChildren(Group2);
contentBranch.addChildren(Group3);
contentBranch.addChildren(Group4);
contentBranch.addChildren(Group5);
contentBranch.addChildren(Group6);
contentBranch.addChildren(Group7);
contentBranch.addChildren(Group8);
contentBranch.addChildren(Group9);
contentBranch.addChildren(Group10);
contentBranch.addChildren(Group11);
contentBranch.addChildren(Group12);
contentBranch.addChildren(Group13);
contentBranch.addChildren(Group14);
contentBranch.addChildren(Group15);
contentBranch.addChildren(Group16);
contentBranch.addChildren(Group17);
contentBranch.addChildren(Group18);
contentBranch.addChildren(Group19);
contentBranch.addChildren(Group20);
contentBranch.addChildren(Group21);
contentBranch.addChildren(Group22);
contentBranch.addChildren(Group23);
contentBranch.addChildren(Group24);
contentBranch.addChildren(Group25);
contentBranch.addChildren(Group26);
contentBranch.addChildren(Group27);
contentBranch.addChildren(Group28);
contentBranch.addChildren(Group29);
contentBranch.addChildren(Group30);
contentBranch.addChildren(Group31);
contentBranch.addChildren(Group32);
contentBranch.addChildren(Group33);
contentBranch.addChildren(Group34);
contentBranch.addChildren(Group35);
contentBranch.addChildren(Group36);
contentBranch.addChildren(Group37);
contentBranch.addChildren(Group38);
contentBranch.addChildren(Group39);

/* This locates the sound file and adds to the scene graph */
addBackgroundSound(contentBranch,new String("file:./Panther.wav"));

/* Add a light to the contentBranch to illuminate the scene */
addLights(contentBranch);
return contentBranch;

/* This uses the an action event to close or exit the game */
public void actionPerformed(ActionEvent e) {
    dispose();
    System.exit(0);
/* This creates a default universe and locale, and creates a window and uses the functions defined in this class to build the view and content branches of the scene graph */

public Maze3D() {
    VirtualUniverse myUniverse = new VirtualUniverse();
    Locale myLocale = new Locale(myUniverse);
    myLocale.addBranchGraph(buildViewBranch(myCanvas3D));
    myLocale.addBranchGraph(buildContentBranch());
    setTitle("Can You Get To The Finish Post?");
    setSize(800, 600);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
            myLocale.removeBranchGraph(buildViewBranch(myCanvas3D));
            myLocale.removeBranchGraph(buildContentBranch());
            setVisible(false);
            exitButton.setVisible(false);
        }
    });
    setVisible(true);
}

public static void main(String[] args) {
    System.out.println("TRY TO GET TO THE 'FINISH' POST");
    System.out.println("Use arrow keys on the keyboard to move around");
    Maze3D skn = new Maze3D();
}

APPENDIX C
Appendix C – Maze Diagram
APPENDIX D
Appendix D - Questionnaire and Results

1. Does the game look interesting?  
   ![Pie Chart: 80% Yes, 20% No]

2. Does it make you want to play again?  
   ![Pie Chart: 70% Yes, 30% No]

3. Is any colour in the game distracting?  
   ![Pie Chart: 0% Yes, 100% No]

4. Does the game look 3D?  
   ![Pie Chart: 90% Yes, 10% No]

5. Is the background sound a good aspect of the game?  
   (Not shown)

6. Does the webpage look motivating?  
   (Not shown)

7. Can you understand everything on the webpage?  
   (Not shown)

Results from Questionnaire
5. Is the sound a good aspect of the game?

- Yes: 0%
- No: 100%

6. Does the webpage look motivating?

- Yes: 20%
- No: 80%

7. Can you understand everything on the webpage?

- Yes: 80%
- No: 20%
APPENDIX E
Appendix E - Screenshots of Game

Starting of game as an application

Finish point of game as an application

Inside the maze as an application

Starting of game in the webpage

Finish point in the webpage

Inside the maze in the webpage