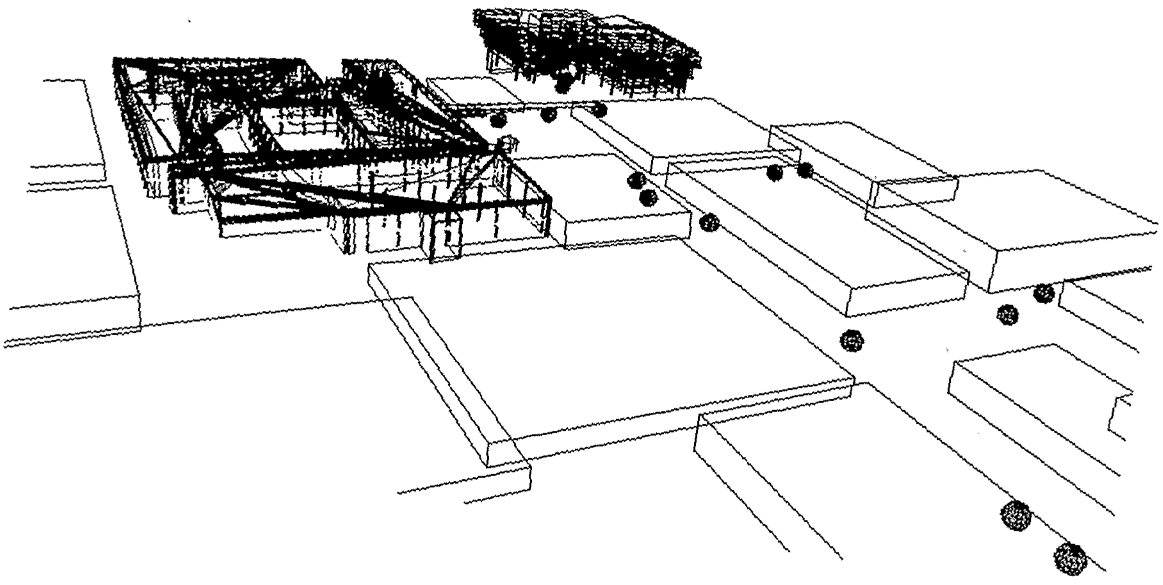
Pathfinding in VRML

Jason Richard Pearce
Edith Cowan University

Pathfinding in VRML



Pathfinding in VRML

by

Jason Richard Pearce

A Thesis

Submitted in Partial Fulfilment of the Requirements for the Award of

Bachelor of Science (Communications & Information Technology) Honours

At the Faculty of Science Technology and Engineering
Edith Cowan University
Perth, Western Australia, Australia

Supervisor: Dr Arshad Omari

Submission date: 5th of January, 1998.

Abstract

Virtual Reality Modelling Language (VRML) is a scene description language which describes three dimensional (3D) space to a computer. Thus the three axis of space that is inherent in our dimension X Y and Z is represented inside a computer. To many people VRML represents a new beginning for the World Wide Web (WWW) because it behaves more like the real world. VRML is experimental, interactive, continuous and of course, three dimensional.

Algorithms in computing have been designed for 2D problem solving and this does not necessarily translate to problem solving on a 3D level. The aim of this project was to experiment with one of these algorithms within the domain of 3D space (VRML).

This project chose to use an algorithm based on solving shortest path problems and then translate this algorithm for it to work in a 3D environment. Java, a programming language, was used to find the shortest path. The shortest path was then shown via the use of an animated camera going through the 3D environment in effect delivering a pathfinding system. VRML was used as the basis to create the 3D environment, thus the project creates a pathfinding system that can be used as a navigational tool within a VRML world.

Declaration

"I declare that this thesis does not incorporate without acknowledgment, any material previously submitted for a degree or diploma in any institution of higher education and that, to the best of my knowledge, it does not contain any material previously published or written by another author or organisation, except where due reference is made, Amen."

Signature: [redacted]

Date: 11-2-98

Acknowledgments

I would like to extend my gratitude to my supervisor, Dr Arshad Omari whose knowledge of shortest path algorithms was critical for the project's development. Dr Omari's patience and understanding throughout the project is also appreciated. Thanks also to the university architect office for the supply of the both the paper and electronic drawings. Lastly, thanks needs to be given to my sister, Karen, for proof reading the work.

Table of Contents

List of Figures

Introduction

The use of 3D computer graphics is becoming increasingly popular, from video games, weather simulations to movies that give us a glimpse of virtual reality. The World Wide Web (WWW) has gained even more popularity for its easy to use interface and its hypermedia abilities that can stretch the globe. Therefore it is natural that people would want to combine the two, marrying the compelling experience of 3D to the global access of the Web. Virtual Reality Modelling Language (VRML) was conceived to join the WWW and 3D graphics together.

VRML is a standard for creating 3D virtual experiences on the World Wide Web. Even though it is in its infancy, VRML solves many of the problems that are associated with typical Virtual Reality systems i.e. cost & portability, while also utilising the enormous distributive power of the Web.

This project seeks to explore the potential of VRML within the context of a pathfinding system that operates within a 3D scene. Pathfinding, in this project, has been defined as solving the problem of finding the shortest path from one point to another. More precisely, in the case of this project, it is to find the shortest possible route from one coordinate in 3D space to another coordinate. The project has investigated the necessities of building a path, based on the 6 degrees of freedom, that is inherent within a virtual world. This pathfinding system has then been incorporated into a 3D map of the ECU campus at Mt Lawley, using VRML as the modeller. Traditionally, pathfinding systems have sought to solve the problem in a 2D environment using the X and Y axis. This project will add the Z axis as well as pitch, yaw and roll. Thus within a sentence this project can be described as,

"To experiment with VRML/Java using pathfinding as an example"

Background to the Study

A pathfinding system in VRML needs to incorporate the background of Virtual Reality (VR), and how this led to the VRML standard. It must also include the background of the WWW, programming languages in VRML, and shortest path algorithms.

Until recently, displaying 3D graphics required powerful computers and so it was limited to research, entertainment and other areas that had large budgets to sustain the need for 3D graphics. 3D computer graphics have always had a niche and so there has never been much need to develop a standard file format for 3D, unlike PostScript with DTP or HTML for Web publishing. Yet with, computer processing power constantly expanding and hardware prices continually falling, 3D content is now becoming accessible for everyone. And there is no better way for distribution of 3D graphics than utilising the already existing network of the Web. In order to utilise the Web, a standard 3D file format was needed and so the VRML file format was conceived. VRML uses the concept of a scene graph to describe the structure of the world, in much the same way that the HyperText Markup Language (HTML) is used to describe the formatting of a document on the WWW. A VRML scene graph can be described using simple words and punctuation (the same as HTML). As such VRML files are written in plain text, and therefore can be created using a simple text editor.

This text file with the suffix .wrl which is sent just like a HTML page (.htm) from a server to a browser. Once downloaded the user needs to have a VRML plugin that reads the .wrl file and converts it into a 3D scene on the web browser. This is the reason HTML experienced such explosive growth. People who were not experts in computers could easily create web pages without the need to use tools specifically for the task. This is also an important attribute for VRML in its early stages, and will ensure virtual reality can be successfully introduced to the general community. This is because virtual worlds will not be created, and should not be created, by computer personnel only, as this would lead to stagnated environments, dominated by a small section of the community, who have the necessary programming skills and software to implement a system.

There is no common definition of virtual reality. Instead the meaning varies, based on the needs of the person using the term. Since the origins of this project borrows heavily from the architectural world, an architectural definition of virtual reality is;

> A technology that enables us to design buildings electronically. These buildings can then be entered , viewed, explored and worked with - all without the use of brick and mortar, or even physical models.
>
> (Jones & Wyatt, 1994, p.468) .

Ultimately though the catch all definition of virtual reality has been described by Hamit (1993, p.11) as;

> A method to allow people to manipulate information in a computer the same way they manipulate objects in nature. It is meant to enhance our ability to deal with the complexities of an increasingly complex society.

In order to utilise VRML for pathfinding a programming language will be needed to calculate shortest paths. By itself VRML is used as a descriptive language of a virtual environment and it is with a programming language that the data describing the 3D world can be manipulated to produce new data that defines the world. Java, a recently devised programming language for the WWW, is the language used for the project.

Search problems are widespread within our world. They deal with the situations in which one choice leads to another and which present problems of organising these choices and choosing the best solution. Shortest path algorithms are designed to solve some of these problems. For many years now, algorithms have been devised to determine the shortest path between two defined points. With VRML, shortest path algorithms can now be introduced to a 3D environment instead of being constrained to the 2D environment.

Significance of the Study

The reason for this project is to look at how VRML and Java can interact with one another using the shortest path problem as the project's goal. There are several good reasons why pathfinding is significant in VRML, a few will be discussed here.

There are occasions when a user of a VR environment does not want to concentrate on manual navigation around the environment. Such an environment is architecture walkthroughs, where visualising the surrounding environment is far more important than manual navigation. Therefore, if the user had the choice of seeing this environment via the use of a animated path, from the users current position to another position of their choice, then the user would gain more out of just visualising the journey, rather than manually controlling the journey.

Pathfinding systems for VRML will allow a user to visualise the journey in virtual reality before undertaking the journey in reality. Current maps describe reality in 2D space. Human beings, however, do not visualise in 2D space but in 3D space. To visually demonstrate the journey in 3D, the user would not remember the name of streets as on a 2D map, but visual signs such as shops, trees etc that are a part of the actual journey. As Sarna and Febish (1996) have emphasised about VRML;

> Adding a GUI is significant because it speaks to your so-called right brain as well as to your left, doubling the brain power that can be applied to understanding an application. Spatial reasoning adds yet another dimension of cognition, making the Internet more human like, conforming better to the way we think and feel.

There is also the use of finding shortest paths for the purpose of data visualisation in 3D. In this project, the shortest path will be shown through the use of an animated camera. However, the shortest path could exist within a data set of coordinates within a 3D space. Such an example would be through the use of network flows or layouts.

Although VRML worlds at the present time are simple and small, as desktop power increases these worlds will become larger and more detailed and eventually, with the aid of VRML 3.0, these worlds can co-exist together, although residing on different machines (i.e. move through a door, which invokes a new world, kept on a server on the other side of the world, without the user knowing). As one industry analyser has expressed about VRML, "the beginnings of tomorrows killer applications are starting to be developed today" (Leinfuss, 1996). Vizard (1996) has emphasised that "VRML has the potential to become an incredibly important IS tool, if used in a innovative way".

There is also the general need to experiment with VRML and to find possible innovative applications for the product. Although most of these experiments may not lead to a direct product, they will add to the bare knowledge base that is known about the potential of VRML.

Pathfinding systems may become invaluable for a new user to the VR environment as a navigation aid in journey visualisation or for possible data visualisation systems where the shortest path in 3D needs to be shown.

Purpose of the Study

For VRML to allow animated paths, it requires the creation of data structures that are unknown in size. One of the drawbacks of current virtual environments is that they are predetermined at the time of their creation. Occasionally an object may change its colour or move to another position, but the data produced is designed to fit in the objects definition i.e. to change a position of an object requires 3 coordinates, these coordinates can be stored within a file.

This project attempts to explore the possibility of using VRML to accept data that is not of a set nature, whose size is unknown and to use the viewpoint of the world as the receiver of this data.

Objectives of the Study

This project has been broken into 3 main components. Two of these are software components the other is theoretical.

Software.

- Development of the VRML model, using the ECU campus (Mt Lawley) as the basis of the model. This model will act as the basis of the pathfinding system, so that the viewpoint will move from the starting coordinate to the end coordinate while avoiding solid objects such as buildings.
- Development of the Java applet, that is used to control the viewpoint through pathfinding, in the VRML model. The applet will act in a kiosk mode with the VRML world.

Theoretical.

- Development of techniques for solving pathfinding problems within a 3D space using the VRML model.

Literature Review

This chapter will discuss the literature relating to VRML, VR and shortest path algorithms. Through this chapter the foundations for the study will be formulated.

General literature

Since the release of the VRML 2.0 specification in August 1996, many books have been released introducing new ideas for the application of VRML. Two very good books widely recognised in the VRML community are Marrin & Campbell (1997) and Ames, Nadeau & Moreland (1997). Marrin & Campbell have illustrated an idea for the use of VRML for a general 3D map modeller;

> A person arrives at a map kiosk in an airport. What if the user had a window on the screen showing the gallery she was standing in at that moment, as if there were a camera in a nearby corner. She knows that her departure gate is building E, so she moves her view of the gallery up and through the roof to see an overview of the airport … She now has landmarks and an approximate walking time as she follows the path she just saw on the computer screen, with no confusion.

To extend this paradigm, it would be far easier if the woman was to simply click an option for building E and the shortest path would be taken from her current position to the destination, instead of having to manually navigate the journey herself. There is also the complete VRML spec 2.0 on the WWW, that was released by the VRML consortium at http://vag.vrml.org.

Virtual reality in architecture is a rapidly expanding area. Architectural walkthroughs can be treated as a well-defined subset of computer-aided design applied to architecture. Watkins & Marenka, (1994) extensively cover the possibilities for VR in architecture by stating that "architects and engineers have long held that they can read blueprints and see the building in three dimensions in their head. Although this may be true, the customers rarely have this gift or training". The authors also emphasises that architects "prefer to visualise their buildings in 3D within their minds rather than looking at 2D drawings".

Vince (1992) has described the usefulness of visualising large architectural and construction projects in 3D and viewing these projects via some aerial route or by taking a car view look of the project. Pathfinding however would allow greater flexibility for the user to describe the path of the aerial or moving car camera shot, instead of the route being fixed beforehand.

Aukstakalnis (1992) describes ways in which VR can be used, although the book does not mention VRML (not conceived at the time of writing). Many examples describing the use of VR however, can now make excellent candidates for VRML.

Since the use of the Java language was necessary for the project, books containing Java information will also be used (Ritchey,1996), (Newman,1996) and (van der Linden,1996). These books, in particular Ritchey (1996), contain ideas about the construction of data structures in Java which was an essential part of the project's development.

Literature on previous findings

Mathematics and computer science have long been involved in solving the problem of finding the shortest path between two points in a graph. By first examining an example (Figure: 1), a better understanding of shortest paths can be established.

Pathfinding system
shortest path = [r 2,2 9,9 10]

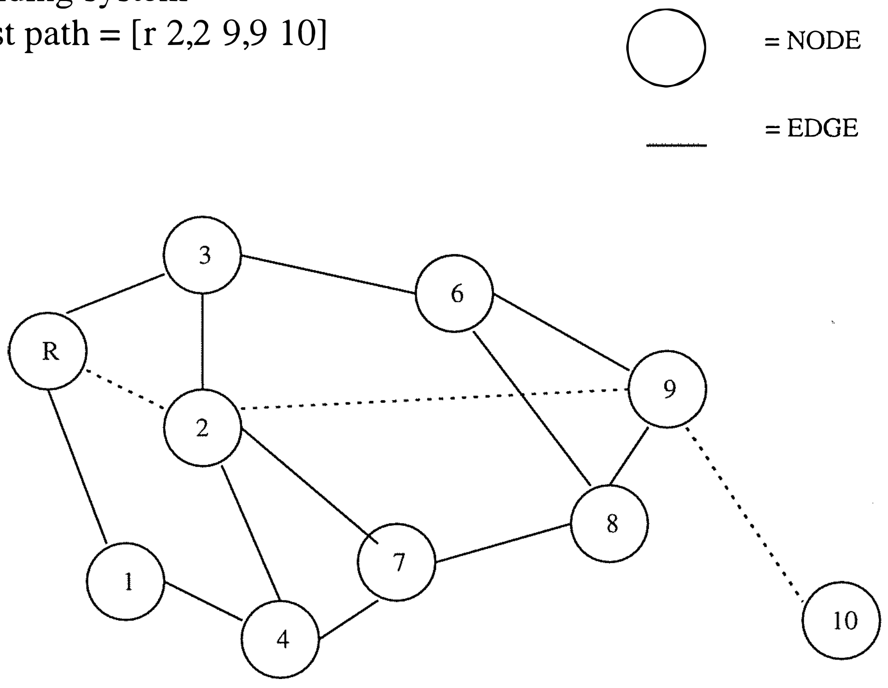


Figure 1: graph (shortest path)

The above diagram is a graph, the graph being a collection of nodes. The graph consists of a distinguished node R, called the root, and zero or more subtrees [R 3, R 2, R 1, R 3 6, R 3 2, etc] each of whose roots are connected by a directed edge to R. The root of each subtree is said to be a child of R, and R is the parent of each subtree root.

Winston (1984) has described several algorithms for finding the shortest path, which can be classified into 2 areas, some path algorithm and optimal path algorithm. Below are a list of algorithms that are separated into these two areas.

Some path - These algorithms are used to find paths from the starting positions to goal positions when the length of the discovered paths is not important, so whether they produce the shortest path is unknown, some examples of these algorithms are;

- Depth first.
- Breadth first.
- Beam.
- Best-first.

Optimal path - These algorithms are more complicated procedures that attempt to find the shortest path, in the most efficient way possible. Except for the British Museum procedure which finds the shortest path simply by comparing all possible paths, some examples of optimal path algorithms are;

- British Museum.
- Branch and bound.
- Dynamic programming.
- A*.

Another method, discussed by Grogono and Nelson (1982) for finding shortest paths is known as backtracking (optimal path solution). In this algorithm, the path is defined from the goal rather than from the root. This backtracking method is built on the conception of problem solving (sometimes its easier to solve a problem by working back from the solution to the question).

A on-line site at http://www.northpark.edu/acad/math/courses /Math_1030/WebBook/ deals extensively with the use of these algorithms and others such as Dijkstra's algorithm and Kruskal's algorithm. These algorithms are elaborated more in (Reingold, Nievergelt & Deo, 1980) ,(Sedgewick,1988) and (Weiss,1993). A more detailed discussion of these shortest path algorithms will be given in analysis and design, and a decision will be made on the eventual algorithm used and why.

It is important to note that Paul Dvorak (1997) has stated that "Problem solving in 2D does not necessarily translate to problem solving in the 3D environment". He elaborates this axiom by explaining that variables in the 3D environment were not considered for the 2D environment because they were not factors. Thus for the project, although the concepts of 2D pathfinding (shortest path algorithms) can be used, additional work must be done in order for the algorithm to work in 3D, and in particular to work in VRML.

Previous studies have shown (Sarna & Febish, 1996) that navigating within a 3D environment on a 2D screen is not easy. Problems are caused by the mixing of metaphors on a 2D environment within a 3D environment. Due to this many different input devices have been used to simulate what is called the 6 degrees of freedom (X, Y, Z axis plus pitch, yaw and rotate) that is inherent within a virtual environment. However, this project's pathfinding system is an easier navigation tool to use because the system handles the navigation from one defined point to another.

Specific studies related to project

Since VRML 2.0's release in 1996, there are now thousands of sites containing VRML worlds across the Web. Many VRML sites are being used to create interesting but not very useful VR worlds. Other projects are attempting to provide useful applications of VRML that can be a source of data visualisation, virtual expos, or virtual cities. These virtual cities and expos may provide 2D maps for the user to use, but again, these, as in reality, can be meaningless because of the difference between visualisation in 2D and 3D.

One ambitious project in its early stages is the world of worlds project by Richard Tilmann (http://www.meshmart.org/wow/). The aim of this project is to produce a integrated virtual map of the major cities of the world. Within these cities, it would initially be useful to navigate around using the pathfinding system for VRML, until the user familiarises themselves with the virtual environment.

Another project features the use of VRML in a virtual expo (http://www.construct.net/projects/ntt/). Again, this virtual expo would have much more user friendliness if the user were able to navigate using pathfinding. Instead the user must explore the site using the mouse or keyboard, which are slow and cumbersome to use.

Other projects more closely related to the study are the use of 3D maps (http://www.bath.ac.uk/Centres/CASA/london). This project is looking at the potential of 3D maps to represent the streets of London and reviewing the advantages and disadvantages of using such maps. Another project (http://cimcentre.snu.ac.kr/~next/viewer/pathfinder.html) is exploring the use of Java in pathfinding on a 2D map ie click two points to discover the distance between the two etc.

Other projects, off-line, include a virtual 200,000 square foot community centre in Los Angeles that was used to illustrate to councillors what the project would eventually look like. The most important aspect of the project was stated by Rick Curoso, the developer, as being, "that the virtual environment allowed the city fathers to participate in the planning process of the development because they could visually understand what the project was trying to achieve" (Bergsman, 1997).

Technology Marketeer, Fujitsu, is at present shopping for retail marketeers to help take its 'virtual mall' by mid 1998 (Matzer, 1996). Although stating that virtual shopping would not overtake reality shopping, Fujitsu believe that this will provide options for users to choose what medium they prefer. The site would allow users to 'walk' through city streets and stores where they can view and purchase products on-line.

There are several universities in the competitive US education market who are experimenting with using VR and VRML to guide prospective students around the campus, showing features the university has to offer. Two such universities are Drew University (Madison, New Jersey), (Briggs, 1996) and UCLA (http://www.gsaup.ucla.edu/vrml/).

There are also many universities in Europe who are beginning to experiment with the applications of VRML within an educational institution. These universities are looking at how, from a tertiary level, VRML can be used as a teaching tool for design in architecture and molecular modelling in chemistry etc. (http://www.rvs.unibielefeld.de/project/vrml-uni/).

Federick Brooks and the University of North Carolina are the pioneers in the field of architectural walkthrough (Watkins & Marenka, 1994). Their work involves the ability to introduce the 'what if' scenario a popular concept in the area of electronic spreadsheets. The 'what if' scenario allows a user to change the look of a building without the need for re-programming. The teams are also looking at methods in which the user can navigate around the virtual environment.

The Seattle Port Authority, in conjunction with the Human Interface Lab in Seattle, is a developing a virtual environment to test proposed expansion plans for the port, again the 'what if' scenario is an important task for this project. Cecil Patterson the IS director of HIL is quoted as saying "there is no substitute for walking around a model in three dimensions to get a feel for how it will do the job" (Watkins & Marenka, 1994). To extend the paradigm for this study, with pathfinding the user would not need to input every step that they wanted to take, instead they could define the path and then just visualise what the journey and the surroundings would look like.

Since the release of SGI's (Silicon Graphic Industry's) EAI (a method of using a programming language to control the world) in early 1997, many projects have begun to create kiosk type interfaces to the 3D world. However as noted by David Brown of SGI;

> There are many examples which use the EAI to create geometry in the scene, and manipulate or animate objects in the world. One important use of the EAI is often overlooked though: using the EAI to control the camera from an applet.
>
> (http://vrml.sgi.com/developer/eai/index.html)

On his paper available at the previous site, David Brown discussed the use of the EAI for creating a Java applet that is used to steer the viewpoint using enhancements such as throttle to give the user more control.

Another project is looking at the control of an underwater vehicle within a 3D space, this project can be found at http://www.stl.nps.navy.mil/~brutzman/dissertation. The object of this project was to control through the use of 3D space, a military underwater vehicle. At the time of writing this project was constructed on a C++ platform however, the writer did express an interest in using the VRML file format because of its simplicity of use and its portability.

To expand this paradigm, another project is looking at the possibilities of using VRML to control processes on the other side of the world http://www.ai.mit.edu/projects/webot/robot/. A robot's environment was translated into VRML, upon which the user could move around to view the robot's environment and make decisions on where the robot should move, instead of being constrained to photographic 2D images of the robot in its reality.

Another closely related project is using VRML in artificial intelligence http://www.coolware.com/lotech/3technology/4ai/. This project, combined with the previous robot project and this project of finding shortest paths within VRML's 3D space and an understanding of the basis of artificial intelligence can be seen, which is an area that the project delves into.

Literature on methodology

Since the structure of the data that needs to be produced is known this project will follow the data-driven design as described by Marco (1979). This methodology is driven by three main principles;

- Determine the structure of the major input and output data streams.
- Describe data structure in terms of sequence, iteration and selection.
- Make design structure conform to structure of data processed.

This method in combination with a prototyping design, where the project is continually re-evaluated based on the results produced by continually improving the prototype, is the basis of the methods that will be used to complete the project.

Summary of Literature Review

VRML is a standard for creating 3D environments. Although still in its infancy it has a great deal of potential in its uses, as shown previously. Shortest path algorithms have been used extensively in computer science for the application of finding the shortest route from one point to another. Through a review of the literature the foundations of the project have been laid, which is to incorporate a shortest path algorithm into the 3D realm of VRML utilising the Java language and to show this path through the use of an animated camera, as though the user themselves was travelling through the path.

Analysis and Design

This chapter will examine the framework upon which the project is based. It is through this theoretical framework that development has taken place.

Choice of development environment

In order for pathfinding in VRML to work a programming language is needed to produce the data structure necessary for the creation of paths. There are two specified methods to utilise languages within VRML, one is internal the other is external. For this project there seemed to be two viable options of using these languages to complete the projects objectives.

- Head Up Display
- Applet - operating as 2D GUI to the world.

A HUD that was actually embedded into the VRML world would use the internal method. The entire code would reside within the .wrl file thus the code would be a part of the VRML scene graph. There are currently three supported languages for this internal method (support for the languages differ from plugin to plugin) these being VRMLScript, JavaScript and Java. The first language VRMLScript was discounted straight away because it was still in its infancy and too simple for extensive work too be done. Javascript was originally the choice to be used in relation to a HUD, however it was soon realised that the language did not have the features required for pathfinding.

This is because the previous two are scripting languages, in that they are created for a specific purpose (simple interactions in VRML and HTML interaction). Thus Java, an object orientated programming language was the choice. However, development of a useable HUD was slow. The issue of designing a scrollable list on a HUD proved to be a real barrier and the project was slowly drifting into the areas that it was not intending to investigate (constructing 3D GUIs). It was at this time that a decision was made to use an external 2D Java applet to control the VRML world, using a kiosk mode metaphor.

Java is an object-oriented programming language developed by Sun Microsystems. Java is compiled to an intermediate byte-code which is interpreted on the fly by the Java interpreter which resides on a client machine. Java programs cannot access arbitrary addresses in memory. This is what makes Java, in theory, able to execute on any platform regardless of its architecture. Java applets can be sent in the same manner that an HTML page is sent on the request of the user. However, as stated before, it is not like HTML or VRML which are description languages. Java is a fully functional object orientated programming language.

This project has created a Java applet which in turn controls the pathfinding aspects of the VRML world through the viewpoint contained within the world.

However, for communication between a VRML world and an external Java applet an interface between the two is needed. This interface is call an External Authoring Interface (EAI) and it defines the set of functions on the browser that the external environment can perform to affect the VRML world. The EAI acts as a layer between the VRML world and the Java applet as shown in Figure 2.

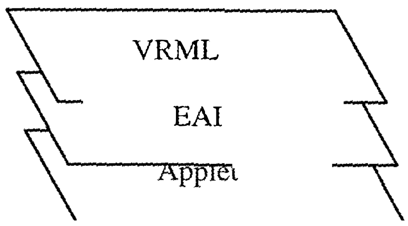


Figure 2: VRML, EAI and Java

A detailed analysis of how this system works will be given later.

The design of this project is centred around the whole project appearing on one HTML page. Within this HTML page their are two embedded objects, the VRML file and the Java class file as shown in Figure 3.

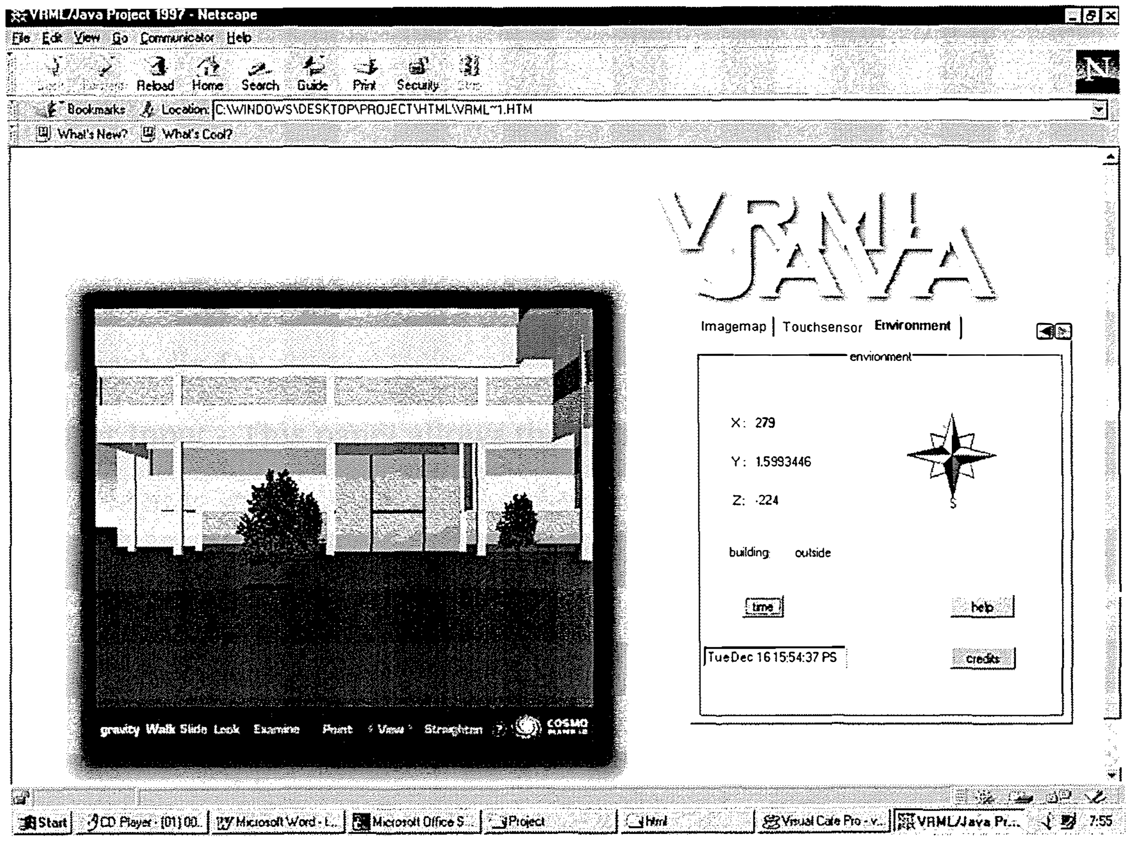

VRML/Java Project 1997 - Netscape
File Edit View Go Communicator Help
Reload Home Search Guide Print Security
Bookmarks Location: C:\WINDOWS\DESKTOP\PROJECT\HTML\VRML~1.HTM
What's New? What's Cool?
VRML JAVA
Imagemap | Touchsensor Environment |
environment
X: 279
Y: 1.5993446
Z: -224
building: outside
time
help
TueDec 16 15:54:37 PS
credits
gravity Walk Slide Look Examine Point View Straighten
COSMO
Start CD Player Microsoft Word Microsoft Office Project Html Visual Cafe Pro VRML/Java Pr... 7:55

Figure 3: VRML and Java applet

The Java applet has been divided into six separate panels, each panel having different functions that can be performed. The following list highlights the name of these panels. More detail on these panels and their functions can be found in Appendix D (user guide).

- Environment - A general panel that is first seen by the user when entering the HTML page as shown in Figure 3. It is used to give the user general information about their place in VRML world.
- Touchsensor - This panel allows the user to find the names of buildings by clicking on the building in the VRML world. A birdseye map of the world in the Java applet highlights the building. (Appendix D).
- Imagemap - This panel uses the same metaphor as above but the user clicks the map in the Java applet and they are teleported in the VRML world to the building entrance. (Appendix D).
- Paths and maps - Similar to the previous panel but instead the shortest path between the user's current position and the destination is shown. (Appendix D).
- User Input - This panel allows the user to enter coordinates and orientation upon submission of these coordinates the user teleports to this position. (Appendix D).
- Teleportation - A list is presented to the user upon which clicking on a building's name in the list, the user teleports to the building's entrance. (Appendix D).
- Pathfinding - Similar to the teleportation panel, instead the shortest path is shown through the use of the camera. (Appendix D).

This whole applet was developed to give the user a sense of place within the VRML world and allow the user to move around the VRML world without the need to navigate with the VRML navigation tools, which at times can be difficult to use. In order to fully understand the project, a brief description on how VRML works will now be given.

VRML basics

In construction, blueprints are used to specify the layout of a building. VRML scene graphs are blueprints for building three dimensional worlds. It specifies and organises the structure of the VRML world.

This scene graph can be described using simple words and punctuation. As such VRML files are written in plain text, so they can be created using a simple text editor. This is the reason HTML experienced such explosive growth.

The three most important components of a VRML scene graph are nodes, fields and routing.

Nodes and fields

There are approximately 60 node types to choose from in VRML. A node in VRML is an element that implements some functionality. The name of the node indicates its basic function ie (Cone, Transform etc). Each node contains a list of fields, which holds values that define the parameters for its function. Below is an example of the syntax of the Cylinder node, bold indicates that it is a part of the syntax for a VRML file.

```
Cylinder
      {
            field SFFloat      radius      1.0
            field SFFloat      height      2.0
            field SFBool       side        TRUE
            field SFBool       top         TRUE
            field SFBool       bottom      TRUE
      }
```

- *Field* - their are five fields that determine the parameters for the function of a cylinder. If none of these fields are used in the syntax of the VRML file then they are set to the default values as shown above. A node's fields has two types of specifiers, type specifier and class specifier.
    - *type specifier* - the type specifier deals with what type of information is stored within the field. SFFloat stands for Single Field Float which is one floating point number which for the cylinder is the type specifier for the radius and height field. The other 3 fields have SFBool type specifiers (holds a value of either true or false).
    - *class specifier* - the class specifier indicates which one of four classes to which a field can belong, these being
        - *field* (static cannot be changed once set),
        - *eventIn* (field can receive events),
        - *eventOut* (field can send events),
        - *exposedfield* (field can send and receive events).

Making a VRML file (.wrl) with just a cylinder node will not do anything. The cylinder node specifies only the geometry of the object, not its appearance. VRML also has an appearance node. Once you have geometry and appearance this can form the basis of a VRML object. All that is needed to do is to associate the geometry with the appearance. In VRML this is achieved with the Shape node. Its definition looks like this:

```
Shape {
                appearance
                geometry
        }
```

Because the type specifier for both of the fields is SFNode (Single Field Node), the fields in these nodes accept other VRML nodes. Thus, the simplest scene graph in VRML might look like this:

```
Shape {
              geometry Cylinder{}
       }
```

This scene describes a cylinder which uses the default values for both its geometry and appearance (2mtr height 1mtr diameter and the colour white) because no data was entered into their fields, this scene in a browser would look like Figure 4.

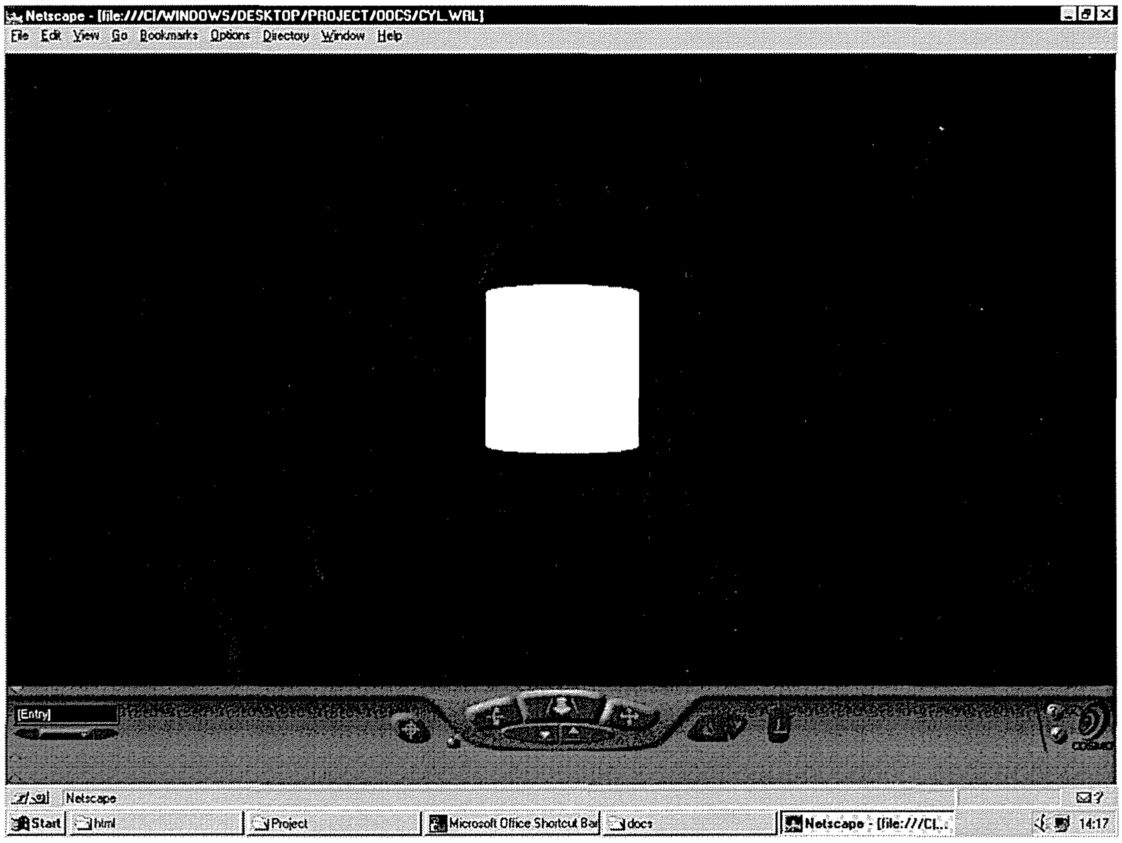

Netscape - [file:///C|/WINDOWS/DESKTOP/PROJECT/DOCS/CYL.WRL]
File Edit View Go Bookmarks Options Directory Window Help
[Entry]
Netscape
Start
html
Project
Microsoft Office Shortcut Bar
docs
Netscape - [file:///C|...
14:17

Figure 4:. VRML basic scene

VRML uses a Cartesian coordinate system. Every point in the world can be described by a set of three numbers, called a coordinate.

These three numbers are represented by X Y and Z. The X component places the object right and left in the world. The Y component places the object up and down in the world and the Z component which places the object nearer and farther from the front of the screen. In the preceding example of the cylinder, because there were no coordinates associated with the Shape node, the shape is placed at the default coordinate 0 0 0.

Coordinates in VRML are defined in units which are metres in the VRML file. In order to place objects throughout the scene, another node must be used to move the shape, the Transform node. So to move the cylinder to the coordinate point of {3,2,2} (3 meters to the right, 2 metres up, 2 metres closer to the viewpoint) the following syntax is required in the VRML file.

```
Transform {
      translation 3 2 2
      children
Shape {
            geometry Cylinder{}
      }
}
```

producing the following result in Figure 5:

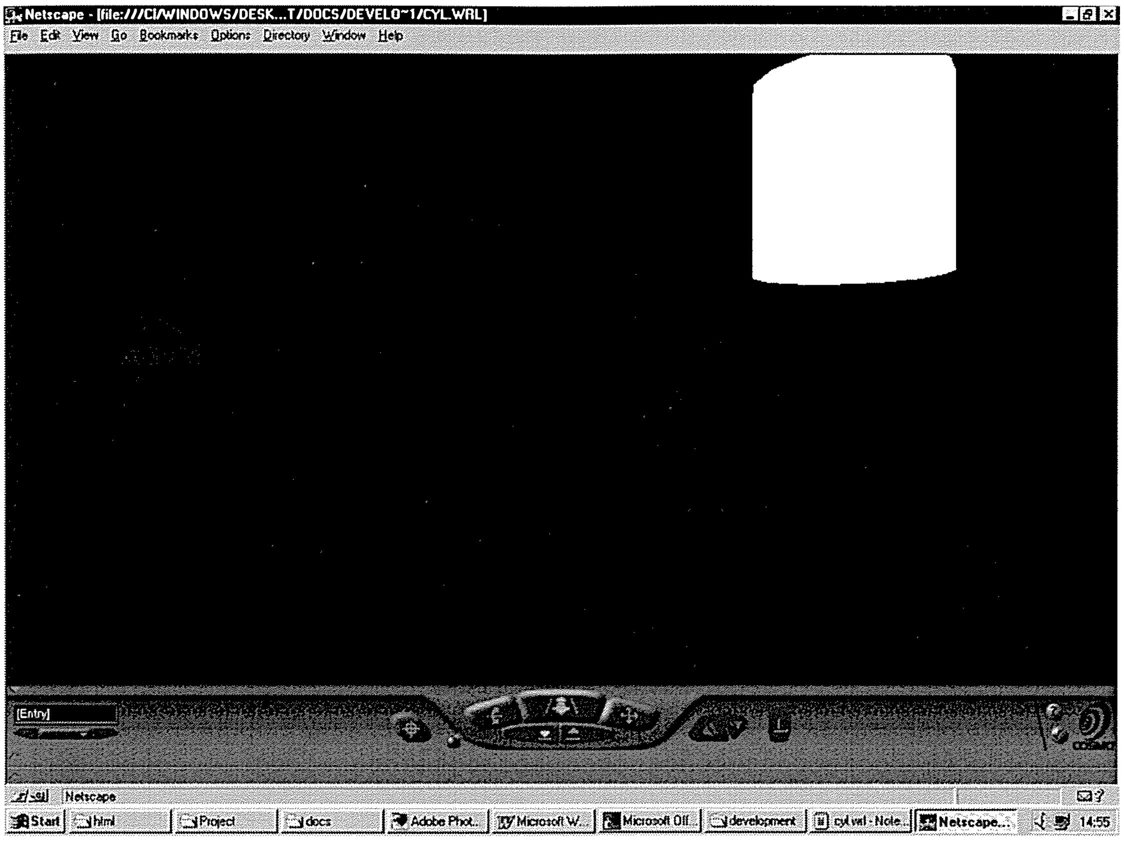


Figure 5: VRML basic scene no. 2

By combining these nodes, one can construct the form that is used to describe the world, So that the following VRML scene would produce the accompanying screenshot.

```
Transform {
      translation 3 2 2
      children Shape
      {
            geometry Cylinder{}
      }
}

Transform {
      translation 1 -2 -2
      children Shape
      {
            appearance Appearance {
                  material Material {
                              diffuseColor 0 0 1
                        }
                  }
            geometry Cone{}
      }
}
```

The above VRML syntax would produce the scene in Figure 6.

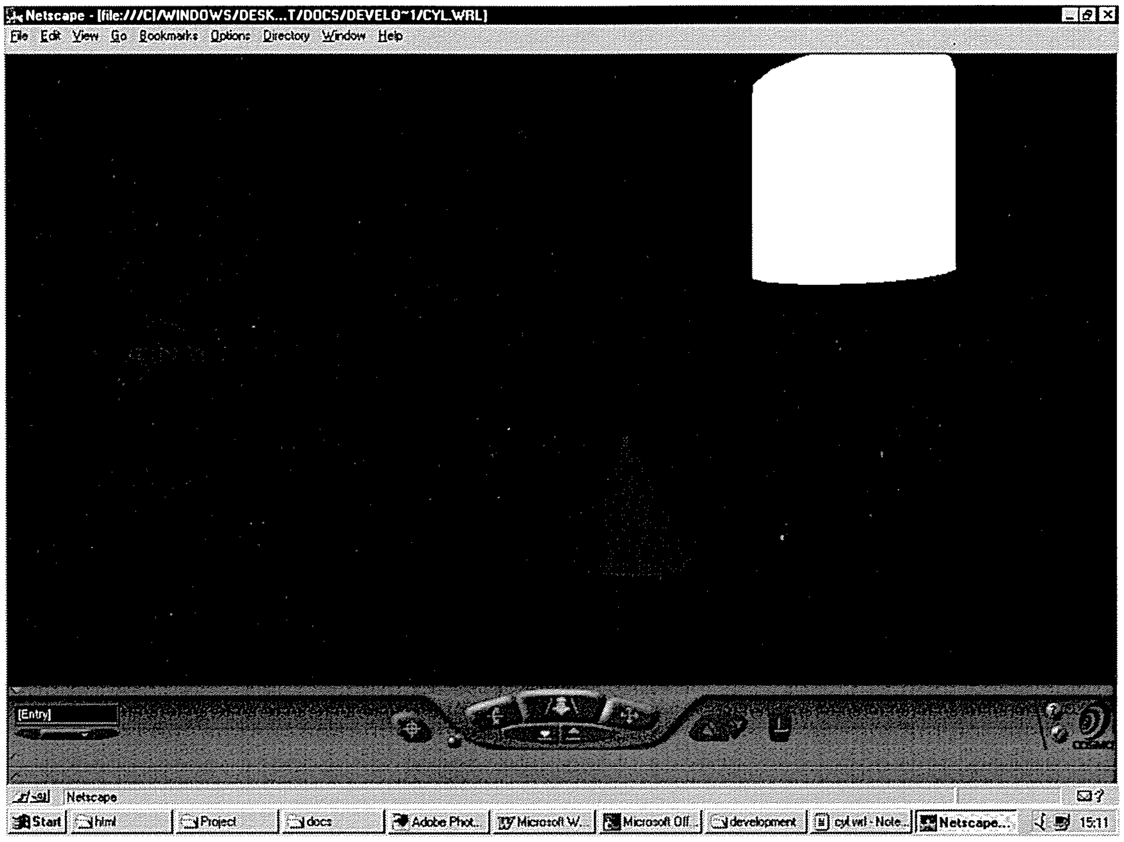


Figure 6: VRML basic scene no.3

By building a scene using these nodes a user would have constructed a scene graph, an example is shown below.

Scene Graph

A simple scene graph for describing a 3D world. This scene graph (Figure 7) is of the previous VRML syntax of the cylinder and cone.

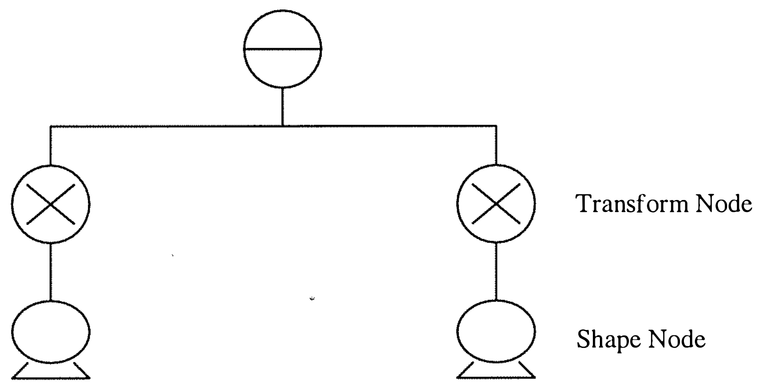


Figure 7: VRML scene graph

VRML and the project

For the purpose of this project it is important to understand the concept of some other nodes; Viewpoint, TimeSensor, PositionInterpolator and OrientationInterpolator.

Viewpoint

The Viewpoint node is used to set up the actual view that the user sees within the VRML world. The following is the Viewpoint node definition;

```
Viewpoint
      {
            eventIn             SFBool       set_bind
            exposedField        SFFloat      fieldofView 0.75398
            exposedField        SFBool       jump        TRUE
            exposedField        SFRotation   orientation 0 0 1 0
            exposedField        SFVec3f      position    0 0 10
            field               SFString     description ""
            eventOut            SFTime       bindTime
            eventOut            SFBool       isBound
      }
```

In the previous examples of VRML syntax, no Viewpoint node was used thus the default Viewpoint was used. The default position for the viewpoint is 0 0 10. To change this the user would need to specify a new position by using the Viewpoint node.

```
Viewpoint {
            position 3 0 7
            #all other fields are still default
      }
Transform {
      translation 3 2 2
      children Shape
      {
            geometry Cylinder{}
      }
}

Transform {
      translation 1 -2 -2
      children Shape
      {
            appearance Appearance {
                  material Material {
                              diffuseColor 0 0 1
                              }
                        }
            geometry Cone{}
      }
}
```

The viewpoint has now moved to the right 3 metres and forward 3 metres from the default shown in the previous screenshot, Figure 6.

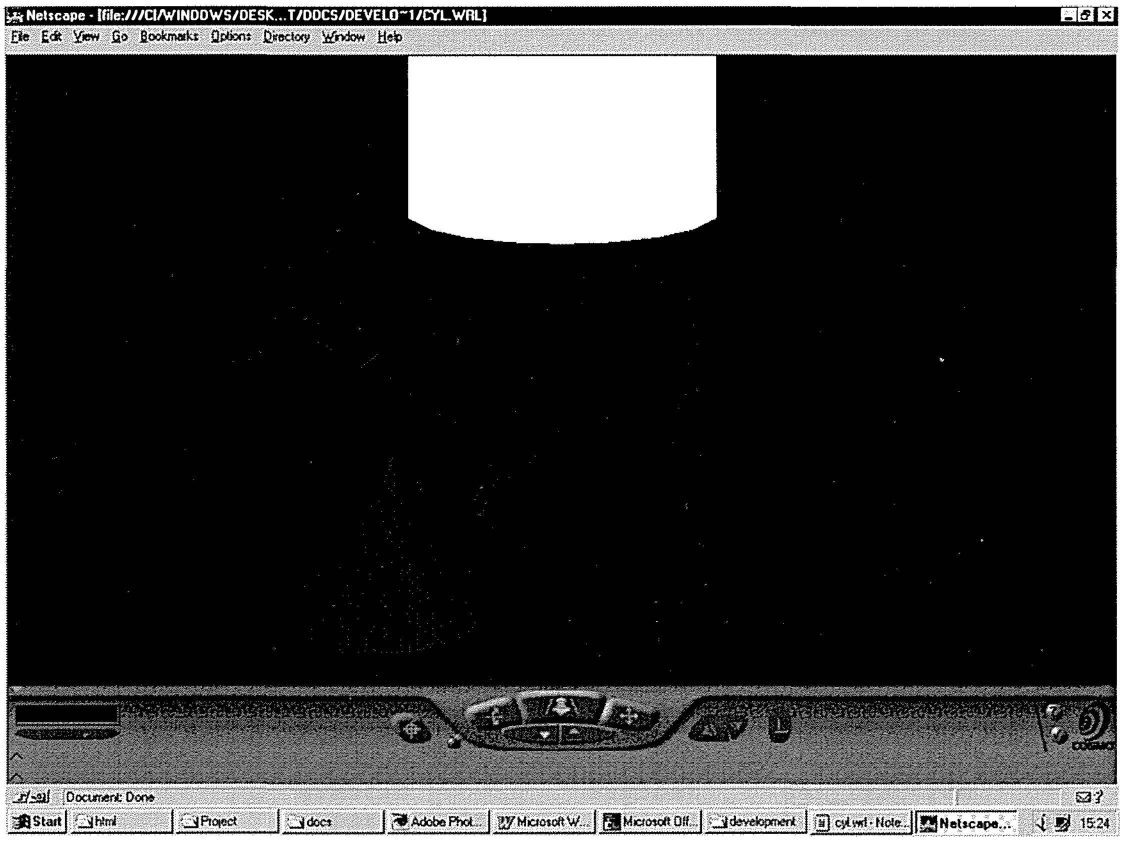


Figure 8: VRML & the Viewpoint

Below is the starting viewpoint of VRML world that was used in the project.

```
DEF CAM Viewpoint {
position 279.1 1.6 -224.1
orientation 0 1 0 1.571
fieldOfView 0.7854
description "Camera01"}
```

The most important fields in the Viewpoint node in regard to the project, is the position and orientation node. The position field obviously defines, in the VRML space, the location of the camera or viewpoint. The orientation field defines the direction that the view is facing, which on a basic level is either north, south, east or west. The 'DEF CAM' syntax is used to give the node a name so that events can be sent to this node to change data in the nodes' fields. Because the object of pathfinding is to produce an animated viewpoint of the path, animation nodes must be used to send new positions and orientations to the Viewpoint node. This is where the PositionInterpolator and OrientationInterpolator nodes are used.

Interpolators

To move the viewpoint from a position of 0 0 10 to 3 0 7 the position of every coordinate between these two coordinates needs to be sent to the viewpoint such as 1 0 9 etc. This is what interpolation achieves. Below is the definition for the PositionInterpolator.

```
PositionInterpolator
      {
            eventIn          SFFloat     set_fraction
            exposedField     MFFloat     key           []
            exposedField     MFVec3f     keyValue      []
            eventOut         SFVec3f     value_changed
      }
```

To illustrate how the interpolator works, some simple values will be added to the key and keyValue fields such as;

```
PositionInterpolator
      {
            eventIn          SFFloat     set_fraction
            exposedField     MFFloat     key          [ 0,     1 ]
            exposedField     MFVec3f     keyValue     [0 0 10,3 0 7]
            eventOut         SFVec3f     value_changed
      }
```

An interpolator node, whether it be for orientation, scale or position, has a list of numeric values called keys and a list of values to interpolate called key values. The type specifier for the key value is determined by the particular interpolator. Every interpolator has a set_fraction eventIn. When this event is received, its value is matched to one of the keys. If a match is found, the key value corresponding to that key is sent out. Thus if the set_fraction receives an eventIn of 0 the match of 0 is made at position 1 in the key and the first position in the keyValue is sent out (value_changed = 0 0 10) and the same with the key of 1 (3 0 7). If a value of .5 is received the formula for the interpolator computes the middle distance between the key values between position 1 and 2 (1.5 0 8.5). By routing the value_changed field to a Viewpoints position, an animated path can be created.

However one thing missing is the value for eventIn field set_fraction, which is the role of the TimeSensor node. In order to understand the role of the Timesensor node in this project the concept of routing in VRML must be understood first.

Routing

In VRML, routes are the wiring that makes animation and user interaction possible. The route command at a basic level looks like this;

```
ROUTE Node1.isActive TO Node2.set_on
```

In this example, the field isActive of Node1 is true when it is activated. This true value is then sent to the field set_on in Node2 therefore set_on is now true. This is much in the same way a light switch is used to turn on a light.

TimeSensor

The TimeSensor node generates time related eventOuts and this is its definition;

```
TimeSensor
      {
            exposedField SFTime      cycleInterval      1
            exposedField SFBool      enabled            TRUE
            exposedField SFBool      loop               FALSE
            exposedField SFTime      startTime          0
            exposedField SFTime      stopTime           0
            eventOut     SFTime      cycleTime
            eventOut     SFFloat     fraction_changed
            eventOut     SFBool      isActive
            eventOut     SFTime      time
      }
```

The most important field in this node is the fraction_changed field. This eventOut generates increasing SFFloat values from 0 to 1 as time proceeds. These values are routed from this field to the interpolators set_fraction, which as discussed before, is used to find the keys and the corresponding key values. Therefore the number of keys = number of keyvalue sets. Also the key field of the interpolator can never go higher than 1, thus all keys represented must be between 0 and 1. Below is an example of how animation works in VRML. In this case the Viewpoint is being animated but any object's position can be animated by changing its position value in its Transform node.

```
DEF CAM Viewpoint {
                        position 0 0 10
                  }

Transform {
      translation 3 2 2
      children Shape
      {
            geometry Cylinder{}
      }
}
```

```
DEF CAMTIME  TimeSensor { loop TRUE  cycleInterval 5 },
DEF CAMPOS PositionInterpolator {
                              key [0,1]
                              keyValue [0 0 10, 3 0 7] },
ROUTE CAMTIME.fraction_changed TO CAMPOS.set_fraction
ROUTE CAMPOS.value_changed TO CAM.set_position
```

Upon loading this world file, the Viewpoint is continually animated (loop = true) from the position of 0 0 10 to 3 0 7 with each loop taking 5 seconds to complete. Routing in VRML is used to send events from one field to another as the last two lines of the previous example illustrate.

Below is the initial set up of the PositionInterpolator and OrientationInterpolator within the VRML world used for the project.

```
DEF CAMTIME  TimeSensor {enabled FALSE}
DEF CAMPOS PositionInterpolator {
                              key [0,1]
                              keyValue [0 0 0 0 0 0 ] }
ROUTE CAMTIME.fraction_changed TO CAMPOS.set_fraction
ROUTE CAMPOS.value_changed TO CAM.set_position

DEF OCAMTIME  TimeSensor {enabled FALSE}
DEF CAMROT OrientationInterpolator{
                              key [0,1]
                              keyValue [0 0 0 0 0 0 0 0 ] }
```

In this project, for both the PositionInterpolator and OrientationInterpolator their are separate Timesensor nodes. Although one could have been used the reason for two will be provided later. Also both TimeSensors are currently turned off ie enabled = false. Via the use of the Java applet, this will be turned on as values are sent to the keyValue field of both the interpolators from the construction of a path.

VRML, EAI and the Java applet

For communication between a VRML world and an external Java applet, an interface between the two is needed. This interface is called an External Authoring Interface (EAI) and it defines the set of functions on the VRML browser that the external environment can perform to affect the VRML world. The EAI described here is designed to allow an external program (referred to here as an applet) to access nodes and its subsequent fields in a VRML scene using the existing VRML event model.

To Java, the EAI is just anther set of classes with methods that can be called to control the VRML world. To VRML, the EAI is just another mechanism that can send and receive events, just like the rest of VRML.

In order to explain how the EAI works with the applet and the VRML world, a simple example of teleportation which was created in the teleportation panel will be used to illustrate the workings of the project.

Teleportation in VRML is achieved by changing the position and orientation values in the Viewpoint node. The Viewpoint node in the VRML file has been defined by the word 'CAM'.

The first thing done in the Java file is to declare the fields that are to be use in teleportation as shown below. Both the eventIn and eventOut of the position and orientation field. Strictly speaking, only the eventIn is needed the eventOut is used to provide the user with information on what the new position and orientation fields now hold.

```
EventInSFVec3f translation = null;
EventOutSFVec3f newtrans = null;
EventInSFRotation rotate = null;
EventOutSFRotation newrot = null;
```

The data types these variables hold are shown by its data type declaration (EventInSFVec3f) which holds 3 floating point values and SFRotation which holds 4 floating point values.

The next thing that needs to be obtained is a reference of the VRML file when the applet is initialized, by using the following code in the init() method of the Java applet.

```
        //getting a hold the browser and embedded file (VRML)
        //this is for netscape browsers only.

    JSObject win = JSObject.getWindow(this);
    JSObject doc = (JSObject) win.getMember("document");
    JSObject embeds = (JSObject) doc.getMember("embeds");
    browser = (Browser) embeds.getSlot(0);
```

As shown before nodes in VRML can be named using the DEF function. Any node that has been defined can be accessed by the applet. Once a pointer to a node is obtained the eventIns, eventOuts (exposedfields have essentially an eventIn and eventOut) of that node can be accessed as shown from the code below from the ecu.java file which compiles to make ecu.class.

```
        //getting a hold of the VRML nodes
        Node camera = browser.getNode("CAM");
```

Once the node has been accessed, the fields of these nodes can be accessed. In this case, the field's name is set_position and not position. This is because the position field is an exposedField and has both a eventIn and eventOut thus these are called set_position and position_changed respectively, although strictly speaking both fields have the same value.

```
        //getting a hold of the VRML fields Eventin contained within
        the nodes
        translation = (EventInSFVec3f) camera.getEventIn("set_position");
        rotate = (EventInSFRotation) camera.getEventIn("set_orientation");
```

```
//getting a hold of the VRML fields Eventout contained within
the nodes
newtrans = (EventOutSFVec3f) camera.getEventOut("position_changed");
newrot = (EventOutSFRotation) camera.getEventOut("orientation_changed");
```

Once this has been achieved i.e. a handle to these two fields has been obtained, new values can be assigned to these fields.

Teleportation

If the user was to double click on building 6 in the list contained in the teleportation panel, the first thing that is executed is the following code.

```
else if (l.getSelectedItem() == "building 6")
      {
       teleportation(b6,east);
      }
```

This enacts the teleportation method sending it 2 arguments, in this case,

```
float b6 [] = {347.5f, 1.6f, -204.7f};
float east [] = {0f,1f,0f,-3.142f};
```

The teleportation method then executes;

```
public void teleportation(float[] telepos, float[] telerot)
      {
            translation.setValue(telepos);
            rotate.setValue(telerot);
      }
```

The data contained within the two arrays are sent to the teleportation method, then they are assigned to the translation and rotate object reference which was shown before;

```
translation = (EventInSFVec3f) camera.getEventIn("set_position");
rotate = (EventInSFRotation) camera.getEventIn("set_orientation");
```

Because this data type is an eventIn the new data is sent to the Viewpoint node defined as 'CAM'. In this case, the entrance of building 6 which is facing in a easterly direction.

This is the basis of how the EAI works. It is simply a tool to allow VRML worlds and Java applets to talk to one another, allowing events to be passed backward and forward to one another.

In order to observe events that are changing within the fields of nodes, another class called the EventOutObserver is needed. The EventOutObserver gets the eventOut of fields from the VRML world. All eventOuts (value of field) need to be referenced first such as;

```
newtrans = (EventOutSFVec3f) camera.getEventOut("position_changed");
newrot = (EventOutSFRotation) camera.getEventOut("orientation_changed");
```

After a reference is made to the eventOut, it has to advise the eventout of the data to the EventOutObserver, in this case using a method declaration;

```
newtrans.advise(this,new Integer(3));
newrot.advise(this,new Integer(15));
```

Then the callback method in the applet has to be overridden, to match the abstract method signature of the EventOutObserver, such as below;

```
public void callback(EventOut who, double when, Object which) {
      Integer whichNum = (Integer) which;
}
```

Then the callback method can be used to observe any changes that are being watched within the VRML file as shown below. Using the teleportation function, the new position and orientation are placed in textAreas for the user to observe.

```
public void callback(EventOut who, double when, Object which) {
      Integer whichNum = (Integer) which;

            if (whichNum.intValue() == 3) {
                  float[] val = newtrans.getValue();
                  float[] val2 = newrot.getValue();
```

```
                    textArea1.appendText("x " +
                    Math.floor(val[0]) + ", " + "y " + val[1] +
                    ", " + "z "+ Math.floor(val[2]) + "\n" );
                    textArea3.appendText(val2[0] + ", " + val2[1]
                    + ", " + val2[2] + ", " + val2[3] + "\n");

          }
     }
```

The shortest path algorithm

Shortest path algorithms differ in complexity, more percisely what differs from one algorithm to another is accuracy against execution time. The more accurate an algorithm is, the longer the computational time it requires. In the case of finding the shortest path, this computational time is taken up in generating alternative paths.

As mentioned during the literature review, shortest path algorithms can be divided into two categories, some path and optimal path.

The some path algorithms just find paths from the root node to the end node. It is the first path that is generated that is defined as the path between the start and the end. Whether this path is the shortest depends upon the structure it is working on and the number of nodes, because they do not compare paths to other paths, they simply find paths. Due to their lack of accuracy, these types of algorithms were not a serious consideration for the project. Although computationally they can be very efficient and quick, it is not evident that they have found the shortest path.

The second type of shortest path is the optimal path, which is when the likely shortest path is found in proportion to the amount of computation that it takes.

The first method to solving the shortest path from one coordinate to another is to build all possible paths and calculate the distance of each path and then return the shortest path. This method will always return the shortest path (accurate) but it is extremely wasteful of execution time, especially for a large group of nodes. In this project 22 nodes are used and it is possible to create 1000's of possible paths. The method is known as the ‘British Museum’ or ‘greedy’ method and was not considered for the project.

The accuracy needed in finding the shortest path, determines the eventual algorithm. A balance must be struck between the need for accuracy and quickness.

The two algorithms considered seriously for the project are the Dijkstra algorithm and the A* algorithm. The Dijkstra algorithm tends towards the greedy side of solving the shortest path, whereas the A* algorithm is computationally more efficient. However, Dijkstra algorithm works best when the nodes of the tree graph are tightly bound together and nodes tend to connect to many other nodes. The A* algorithm is most accurate when nodes are relatively spaced apart with a smaller number of connections. As the eventual higher order data structure within the VRML world would fit this second description, the A* algorithm was used.

The structure of how the A* algorithm performs is shown in Figure 9.

```
To do A* with lower bound estimates:
    1. Form a queue of partial paths. Let the initial queue consist of
       zero-length, zero-step path from the root node to nowhere.
    2. Until the queue is empty or the goal has been reached,
       determine if the first path in the queue reaches the goal node.
        2a. If the first path reaches the goal node do nothing.
        2b. If the first path does not reach the goal node:
            2b1. Remove the first path from the queue.
            2b2. Form new paths from the new path by extending
                 one step.
            2b3. Add the new paths to the queue.
            2b4. Sort the queue by the sum of distance
                 accumulated so far and a lower bound estimate
                 of the distance remaining, with least distance
                 paths in front.
3. If the goal node has been found announce success, otherwise
   announce failure.
```

(Winston, 1984)

Figure 9: A* algorithm

How this algorithm was implemented will be shown in the development chapter.

Development

Construction of a higher order data structure

In order for pathfinding to work within VRML, a higher order data structure had to abstracted from the VRML world and made accessible in the Java applet. VRML deals in form but what was needed were coordinates in the VRML world to be used as nodes in the construction of paths in the Java applet. In this case, the structure had to be a set of coordinates extracted from the VRML model as shown in Figure 10.

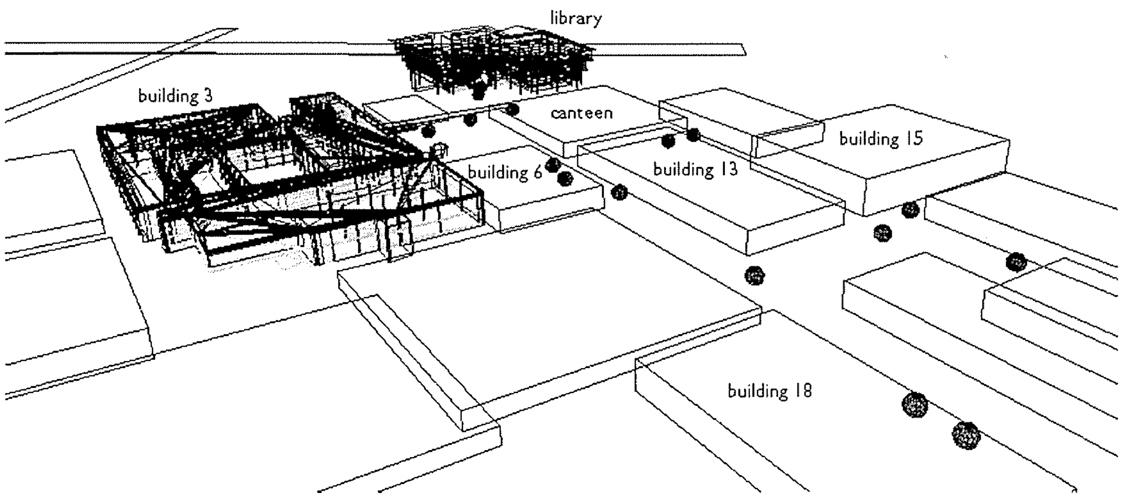


Figure 10: Coordinates from the VRML world

This screenshot of the world file used in the project (shown only in wireframe to indicate the coordinates) shows 22 white spheres (not all can be seen). These spheres, which are only shown here for representation purposes (they are not a part of the ecu.wrl file) represent coordinates in the VRML world, with most of the coordinates being entrances to buildings. The other coordinates are special nodes used to connect other nodes together.

This is because for pathfinding in VRML to work, nodes have to be 'line of sight' to construct paths. Remembering that the PositionInterpolator uses a linear interpolation between two points ie a straight line from the coordinate of building 6 can not go directly to the canteen because it requires it to go around a corner thus the point in the middle that is not outside of any building is called n1. All other points are named after the building that they are associated with (guild, b3, library etc). To get a better understanding of how these coordinates can form a graph so that shortest paths can be found, a view from the top of the ECU world needs to be shown as in Figure 11.

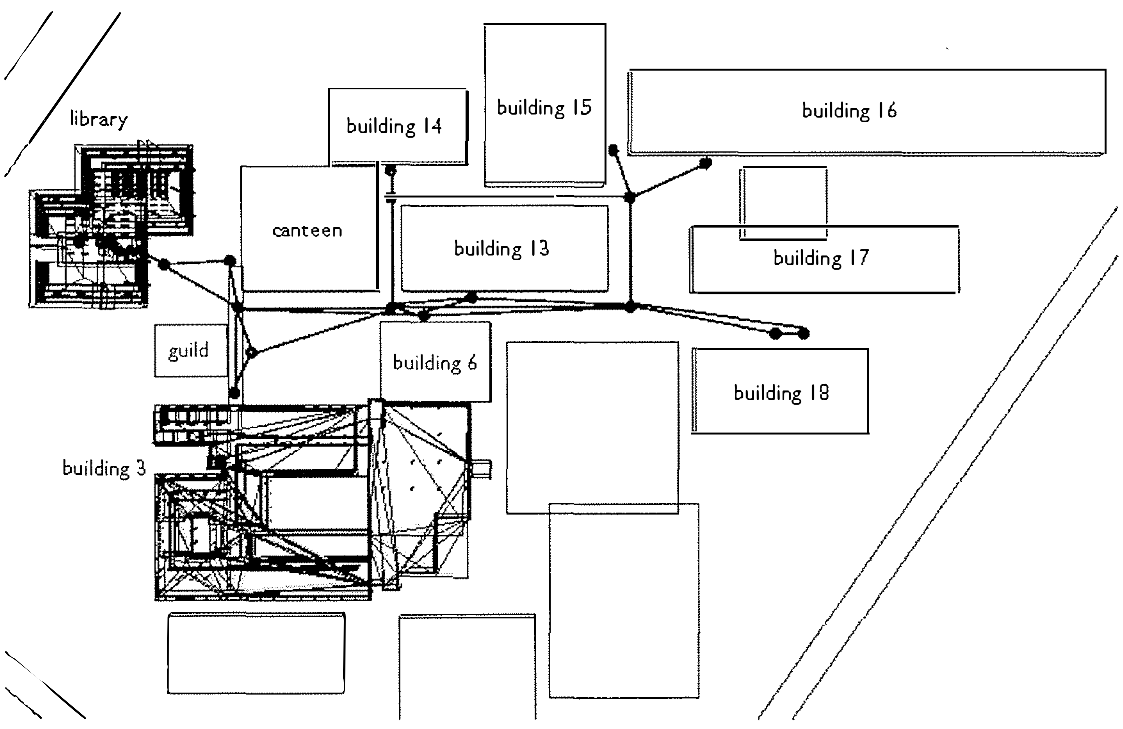


Figure 11: Top of the VRML world

From this screen shot the basis of pathfinding can begin. Nodes are represented by coordinates from the VRML world and edges are constructed on the basis of whether the coordinate can connect with another coordinate without it going through something solid like another building. In order to illustrate this structure in more detail an example will be used on a smaller area containing only 7 coordinates as shown in Figure 12.

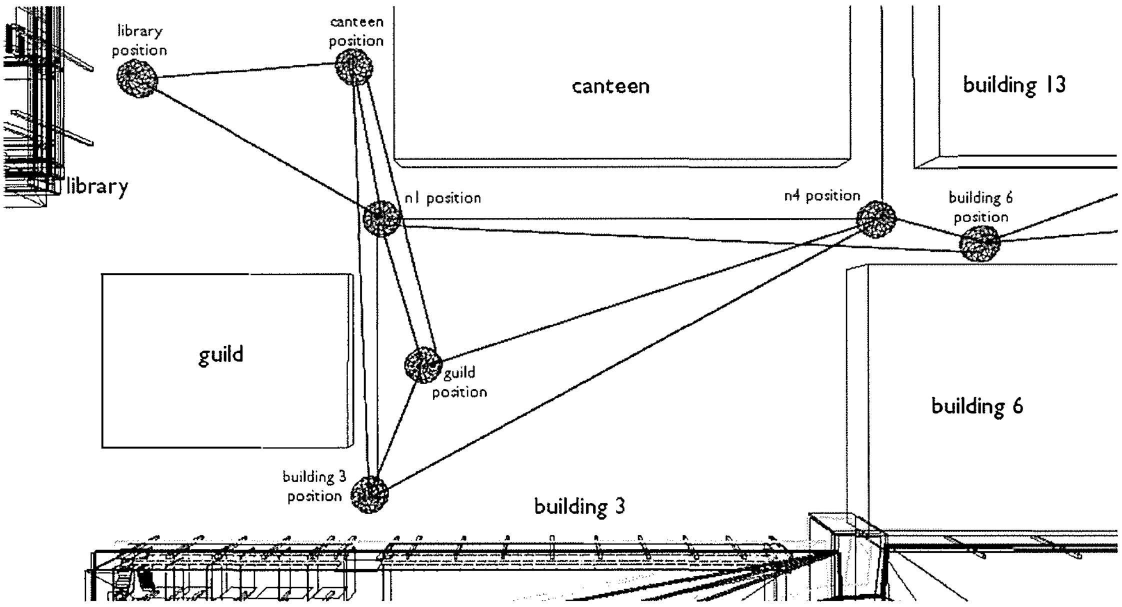


Figure 12:Top of VRML with just 7 coordinates

This screenshot with the coordinates can be represented in a 2 dimensional diagram in the following way;

Diagram of a 2D representation of the VRML world (Figure 12).

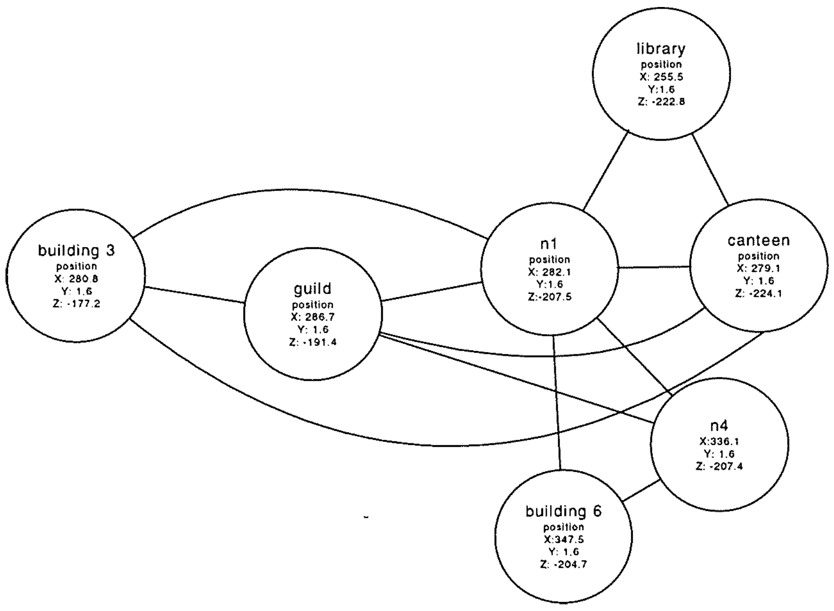

library position X: 255.5 Y:1.6 Z: -222.8
building 3 position X: 280.8 Y: 1.6 Z: -177.2
guild position X: 286.7 Y: 1.6 Z: -191.4
n1 position X: 282.1 Y:1.6 Z:-207.5
canteen position X: 279.1 Y: 1.6 Z: -224.1
n4 X:336.1 Y: 1.6 Z: -207.4
building 6 position X:347.5 Y: 1.6 Z: -204.7

Figure 13:2D graph representation of figure 11

As can been seen, the basis of pathfinding relating back to how shortest paths work, can now be seen. Whether coordinates are also elevated in the world, like the first floor of the library does not matter because this is just another node which is connected only to the node at the front of library and obviously to the second floor of the library. It has to be stated though in order to go from the first floor to the second floor intermediate coordinates are used to go up the stairs. The whole representation of the points in 3D can be flattened into a 2D diagram, which enables the structure to be represented within the Java applet. As Figure 14 shows.

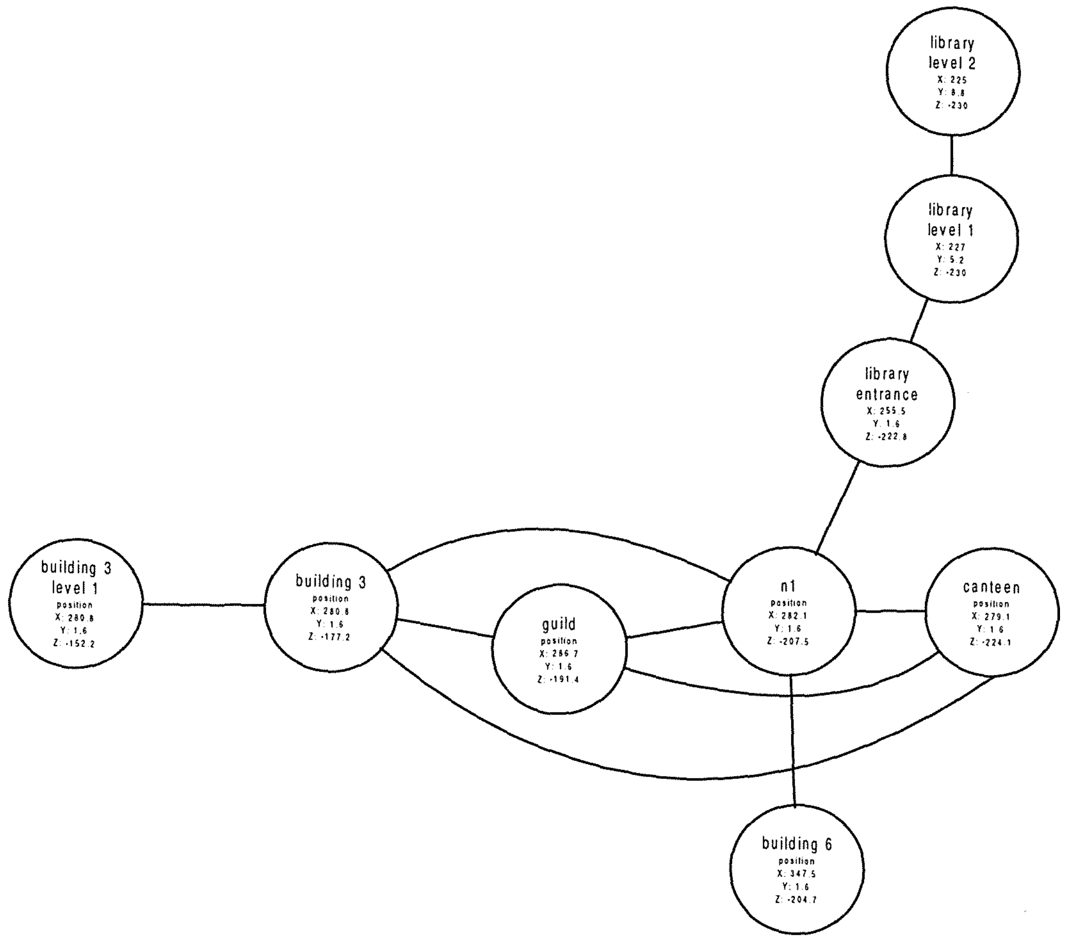

library level 2 X: 225 Y: 8.8 Z: -230
library level 1 X: 227 Y: 5.2 Z: -230
library entrance X: 255.5 Y: 1.6 Z: -222.8
building 3 level 1 position X: 280.8 Y: 1.6 Z: -152.2
building 3 position X: 280.8 Y: 1.6 Z: -177.2
guild position X: 286.7 Y: 1.6 Z: -191.4
n1 position X: 282.1 Y: 1.6 Z: -207.5
canteen position X: 279.1 Y: 1.6 Z: -224.1
building 6 position X: 347.5 Y: 1.6 Z: -204.7

Figure 14: 2D representation of 3D scene

Higher order data structure in Java

The basis of pathfinding in VRML requires the following information being available in the Java applet;

- The coordinates in space.
- What coordinates are connected to what coordinates (edges).
- What is the distance between these connected coordinates (the weight of the edges).

To illustrate shortest paths, an example of just 7 coordinates (guild, library, building 3, canteen, n1, building 6, n4) will be used, although in the final project 22 coordinates were used. The first thing that was stored in the Java applet were the coordinates in the VRML world. This was achieved by using a multiple subscripted array, in this case 6 rows (the number of coordinates) by 3 columns storing the X Y and Z of a coordinate, as shown below the array is declared;

```
float points [][] =  {{286.7f, 1.6f, -191.4f},//guild
                      {255.5f, 1.6f, -222.8f},//lib
                      {280.8f, 1.6f, -177.2f},//b3
                      {279.1f, 1.6f, -224.1f},//can
                      {282.1f, 1.6f, -207.5f},//n1
                      {347.5f, 1.6f, -204.7f},//b6
                      {336.1f, 1.6f, -207.4f}};//n4
```

The second piece of information that needs to be declared is what coordinates connect to what coordinates i.e. does the library connect to building 3 etc. This is achieved again by using a multiple subscripted array determined by the number of coordinates by the number of coordinates (7x7).

```
float edges [][] =    {{0,0,1,1,1,0,1}, //guild
                       {0,0,0,1,1,0,0}, //lib
                       {1,0,0,0,1,0,0}, //b3
                       {1,1,1,0,1,0,0}, //can
                       {1,1,1,1,0,1,1}, //n1
                       {0,0,0,0,1,0,1}  //b6
                       {1,0,0,0,1,1,0}}; //n4
```

Taking the first row which reads 0 0 1 1 1 0 1, this row determines what is connected to the guild coordinate thus;

0 (first column) - indicates that the guild is not connected to itself because the guild is row number 1 in the array.

0 (second column) - indicates that the guild is not connected to row 2 which is the library.

1 (third column) - indicates that the guild is connected to building 3.

1 (fourth column) - indicates that the guild is connected to the canteen.

1 (fifth column) - indicates that the guild is connected to n1.

0 (sixth column) - indicates that the guild is not connected to building 6.

1 (seventh column) - indicates that the guild is connected to n4.

One piece of information missing is the weight of each of the edges (distance), which is completed on the fly as the pathfinding function is executed. Distances could have been placed within the array instead of just using a 1 to show that it is connected, but this would have meant manually calculating each distance and placing it within the array, a procedure that may have introduced errors and is not portable.

The following array was used for purposes of relating information to the previous two arrays, in that row 1 in all the arrays gives information about the guild, i.e. where it is, what is connected to it and what is its character ('a'). This array uses characters and thus is limited to 127 coordinates in the VRML space, but it could have easily accommodated string values, for purposes of simplicity characters were used only.

```
char label[] = { 'a' /*guild*/,
                 'b' /*lib*/,
                 'c' /*b3*/,
                 'd' /*can*/,
                 'e' /*n1*/,
                 'f' /*b6*/
                 'g' /*n4*/};
```

To solve the problem of finding the shortest path from one coordinate to another in the VRML world, the problem was broken down into four smaller functions. The first function solves the nearest neighbour problem. The second function finds the actual path, while the third function converts the shortest path into coordinates and orientations. The last function is that these coordinates and orientations are then sent in pairs to their respective interpolator until the viewpoint arrives at its destination. These functions will be explained in more detail in this chapter.

In order to better understand these functions the following scenario will be used. The user has entered the VRML world and has moved manually to a position somewhere between the canteen and the library (Figure 15).

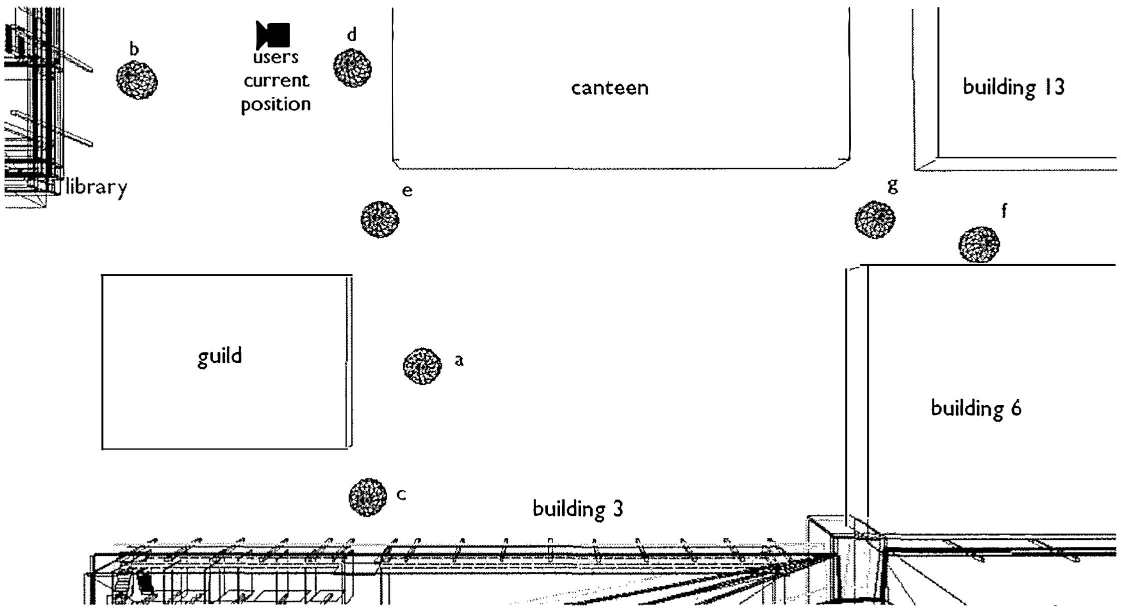


Figure 15: User's current position and orientation in VRML world.

Given this the user wishes to go to the entrance at building 6 'f' and has clicked on the building's name in the pathfinding panel of the Java applet.

Initialisation of the pathfinding

When the user clicks on a destination two things happen. The first is that two arguments are sent to the nearest neighbour method. In this case because the destination is building 6, the arguments ‘f’ (character of building 6) and the integer ‘5’ (the row number regarding information on building 6, its coordinate and what is connected to it, remembering that 0 is row 1) are sent and this is the endpoint of the path. Also three variables are set up (X Y Z) to identify the end of the animated path and thus turning it off through the use of the EventOutObserver, but this situation does not happen till the end where it will be explained in more detail.

Nearest neighbour

The nearest neighbour function only knows one thing at execution and this is the destination coordinate (node). The function of nearest neighbour is to find the user's current position and from this find the nearest coordinate in the points array (data structure) that the user is closest to.

Within VRML a user could be anywhere, thus the first problem to be solved is to find where the user is (Viewpoint) and find the nearest coordinate (neighbour) to the user's current position (put the user on the data structure). The user's current position is obtained by using a global proximity sensor. A proximity sensor node in VRML sends out events when the user enters or exits a defined region of the scene. While inside this region, it reports the users location during movement. Thus by using this node that encompasses the whole ECU world, the user's present position can always be traced. The proximity sensor used in the project is shown below.

```
DEF GLOBALPS ProximitySensor {
                                    size 1000 66 1000
                              }
```

A field not shown here is the position_changed field which has a class specifier eventOut. This field relates the position of the Viewpoint inside the ECU world. This eventOut is routed to the Java applet as a SFVec3f type specifier which is 3 floating points for X, Y and Z.

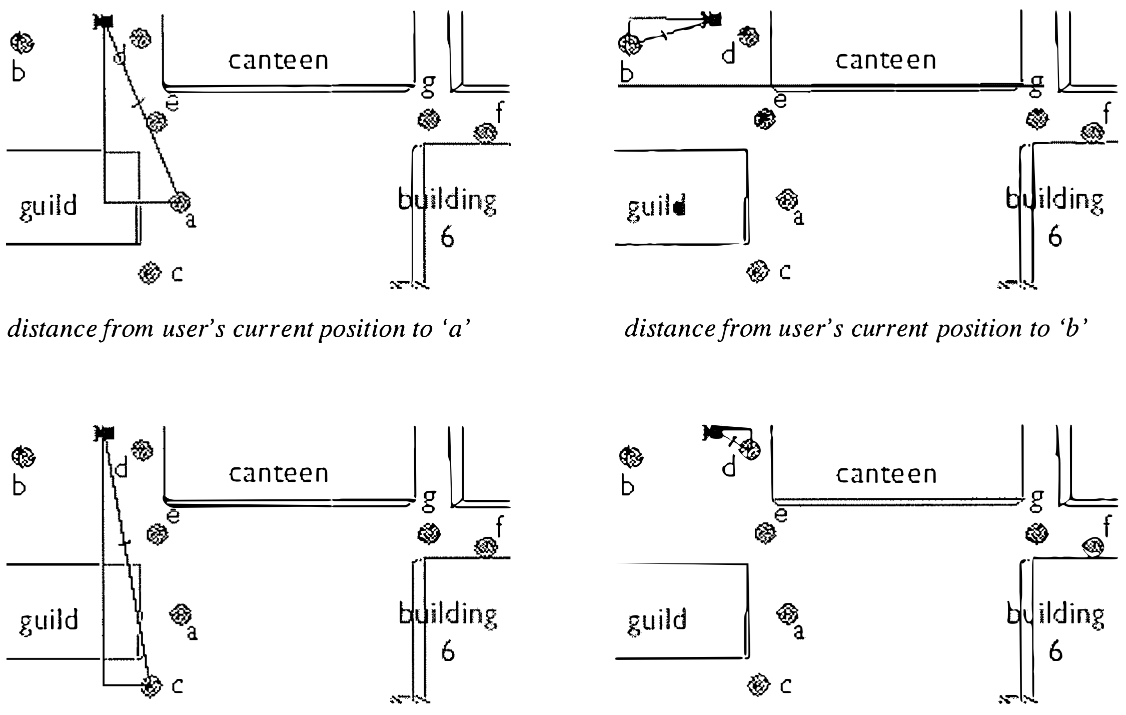
Once the user's current position has been obtained, it can then be compared to all the other coordinates in the points array which is a part of the higher order structure, as the following pseudocode illustrates.

```
nearest_neighbour:
GET endnode_character
GET endnode_index
GET users_current_position
    DOWHILE x =0  x <= number_of_coordinates x++
        SET distance TO users current_position - coordinate_position
        IF distance < lowest_distance
            SET lowest_distance to distance
            SET index TO index_of_closest_coordinate_position
        ENDIF
    ENDDO
ADD character[index] TO root_path
IF character[index] != endnode_character
    findpath
ENDIF
```

Figure 16: Pseudocode for nearest neighbor function (Java source code pg132)

The user’s current position is mapped to the closest node within the higher order structure or the points array.

Thus this section takes the users current coordinate from the proximity sensor and by looping through every coordinate calculates which coordinate is the closest to the user simply by using pythagoras, as shown in Figure 17, for the first four loops.



distance from user's current position to 'a' *distance from user's current position to 'b'*

distance from user's current position to 'c' *distance from user's current position to 'd'*

completed for all other nodes, 'e', 'g' etc

Figure 17: finding nearest neighbour

Once the closest coordinate has been found, the index of this coordinate from the points array is placed within a variable. This is the index of the path to be extended and it is also the root node within the label array. This root node is then placed within a queue. As the user's position, which was abstracted from the proximity sensor is X: 277.1 Y: 1.6 Z:-226.1, then the canteen coordinate would produce the lowest distance to the users current position. Thus the queue would read [d], the character that relates to the canteen.

At the end of this function, the root node [d] is compared to the end node 'f', to see whether the root node is also the end node. If it is not, the findpath function is executed.

It was decided during this stage that the user's position should also map to the same level. This is why the Y point was not considered in the calculation. Although a user could be closer to a node on a higher or lower level, it was decided that the root node should be mapped to the same level that the user was currently in.

One weakness of this function is its inability to know whether a nearest node is on the opposite side of a wall or a building. Minor rework could not fix the problem as it required a completely different method of solving the problem using different types of information. For example, all form in the world would need to be extracted into a database and straight line comparisons would need to test whether the nearest node intersects with a solid object. However because the system has been set up for line of sight between the nodes this problem is greatly reduced.

Finding the shortest path

The findpath function has two pieces of information that it knows, it can tell what the starting coordinate on the data structure is and what the end coordinate is, the result being that findpath performs, the function of finding the shortest path between the start and end. Given that, this is the following pseudocode for the findpath function.

```
findpath:
SET coordinate_extend TO coordinate[index]
SET coordinate_end TO coordinate[endnode_index]
DOWHILE x=0 x<=number_of_coordinates x++
        IF edge[index][x] >=1
                SET path TO root_path
                        IF last element in path != endnode
                                ADD edge TO path
                                SET estimate_distance_of_path TO
                                    sum_of_distance_so_far + distance_to_go
                        ENDIF
                        IF estimate_distance_of_path < lowest_path_distance
                                SET lowest_path_distance TO estimate_distance_of_path
                                ADD path TO front of possible_paths
                        ELSE
                                ADD path TO possible_paths
                        ENDIF
        ENDIF
ENDDO
SET check_path_if_end TO first element of possible_paths
        IF check_path_if_end last character == endnode
                javatovrml
        ELSE
                SET root_path to first element of possible paths
                SET lowest_path_distance TO 9999
                findpath
ENDIF
```

Figure 18: Pseudocode for A* procedure (Java source code pg. 133)

The first thing this function does is extend the root node [d] to see what is connected to the root node, this is where the edges array is used, 1 means there is a connection, 0 means there is no connection.

By scanning through the edges array at row d the fourth row, the loop checks to see what is connected to the canteen coordinate. The first connection is the guild coordinate, thus the queue now reads [d,a]. Then the distance travelled so far from d to a is calculated and is added to the distance remaining from a to f (as the crows flies) as the following Figure illustrates.

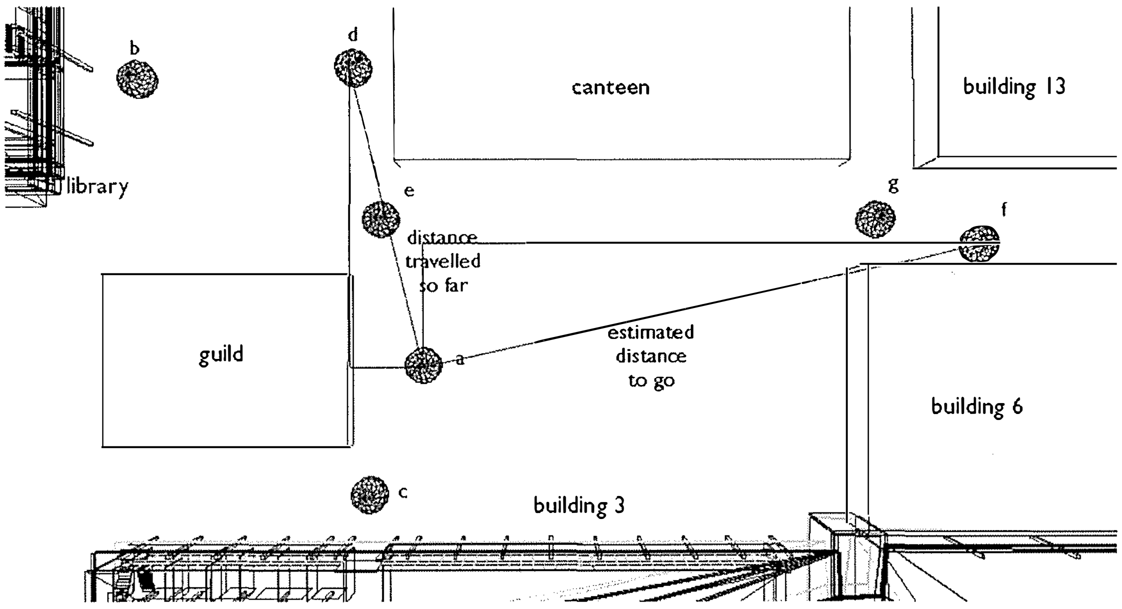
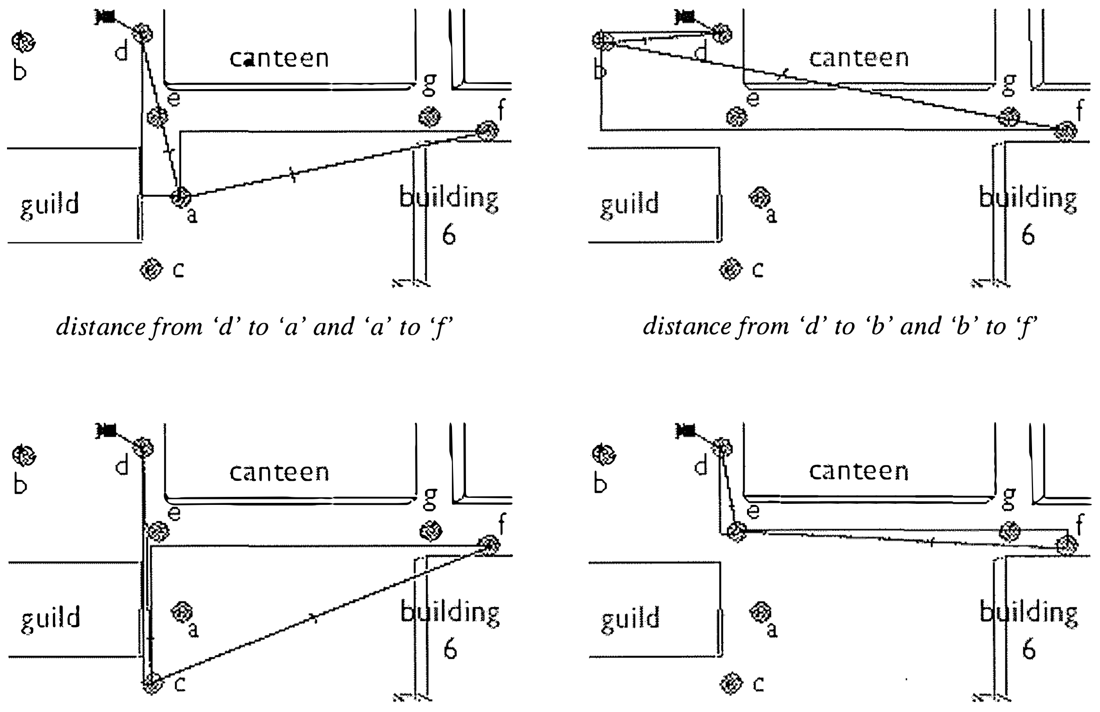


Figure 19: finding the path from the root node to the end node

This distance gives a rough estimation on the total length of the path.

$$\begin{array}{rl} & \text{Distance travelled so far} \\ + & \underline{\text{Estimated distance to go}} \\ & \text{Estimated distance of path} \end{array}$$

This is completed for all possible connections to the d node.



distance from 'd' to 'a' and 'a' to 'f'

distance from 'd' to 'b' and 'b' to 'f'

distance from 'd' to 'c' and 'c' to 'f'

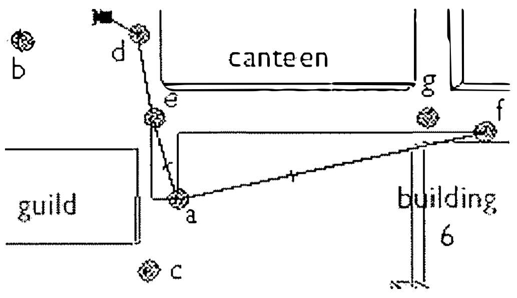
distance from 'd' to 'e' and 'e' to 'f'

Figure 20: All connections from d are made and compared

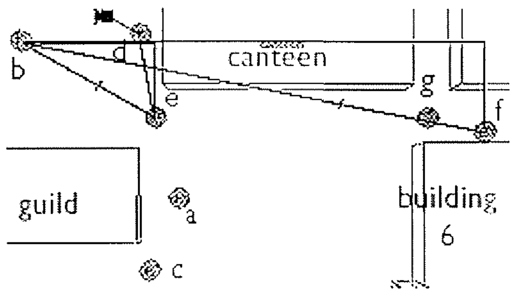
The partial path with the lowest estimated distance is placed at the front of a queue, hence the queue would read as the following;

[[d,e],[d,a],[d,b],[d,c]]

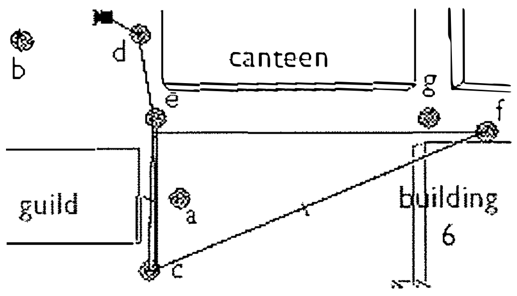
The function then checks to see whether the last character in the first path from the queue [d,e] in this case 'e' is the destination 'f', if it is then the java to vrml function is executed otherwise the index of the coordinate is updated for the last character in this case e = 4 (the fifth row in the edges array). The whole process then starts again with [d,e] being the root node which could be called now the root path and all connections to 'e' are tested, these connections are shown in Figure 21.



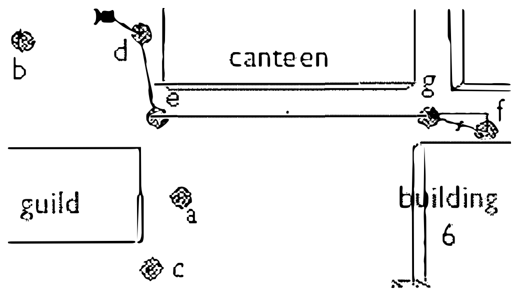
distance from 'd,e' to 'a' and 'a' to 'f'



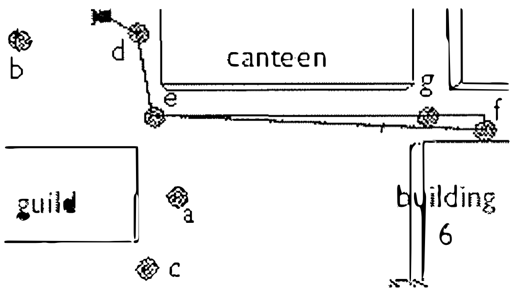
distance from 'd,e' to 'b' and 'b' to 'f'



distance from 'd,e' to 'c' and 'c' to 'f'



distance from 'd,e' to 'g' and 'g' to 'f'



distance from 'd,e' to 'f' and 'f' to 'f'

Figure 21: All connections from e are made and compared

The algorithm also checks to see whether paths that are being extended already contain a visited node such as the situation as [d,e,d], as such this path is ignored. This can occur if the destination is on the other side of the building and the algorithm keeps turning on itself. Also by doing this, it cuts down the time it takes to find the path by dropping paths that could not be the shortest path.

The whole process in this example ends with [d,e,f] being at the front of the queue and this is the shortest path, now this path has to be converted into coordinates for the VRML world's Interpolators thus the javatovrml() method is executed.

Java to VRML

From this section, the path which is the first queue in the master queue needs to be converted into a set of coordinates for the keyValue field within the PositionInterpolator and into set of orientations for the OrientationInterpolator. The pseudocode for this section is shown below.

```
java_to_vrml
SET shortest_path TO possible_paths(first element)
GET users_current_position
GET users_current_orientation
DOWHILE x=0 x<= number_of_characters_in_shortest_path x++
    SET path_character TO current_character_in_shortest_path
    DOWHILE x=0 x<= number_of characters_in_label x++
        IF label_character == path_character
            SET position_array TO coordinates[label_index]
            SET orientation_array[first 3 postions in each row]
        ENDIF
    ENDDO
ENDDO
DOWHILE x=number_of_elements_in_path x<= number_of_elements_in_path -2 x++
    SET orientation_array[3] TO radian_angle
ENDDO
positioninterpolator
```

Figure 22: Pseudocode for java to vrml function (Java source code pg. 134)

By extracting the labels one by one off the shortest path, the index of these characters within the label array points to the coordinates within the points array. These coordinates are then placed within another array which will be dispatched to the VRML file.

Thus the array that holds all of the coordinates needed for the path would read as follows;

```
[[277.1, 1.6, -226.1], //current position  proximity sensor
 [279.1, 1.6, -224.1], //nearest neighbour 'd'
 [282.1, 1.6, -207.5], //shortest path     'e'
 [347.5, 1.6, -204.7]] //destination       'f'
```

This is the set of coordinates needed by the PositionInterpolator in order to route its information to the Viewpoint starting position to the end position. The first coordinate is the user's current position obtained from the ProximitySensor and the nearest coordinate the second coordinate, is the canteen coordinate, the last two are the coordinates for n1 and building 6 (current position of user + [d,e,f]).

However the calculations for the orientation of the Viewpoint have not been evaluated yet, but now the coordinates have been obtained the orientation of the Viewpoint can be completed.

So far in this project discussion on VRML has been based on coordinates ie the position field in the Viewpoint and the key value field in the PositionInterpolator that have a X Y Z value within the Cartesian coordinate system. Orientation fields though hold four values.

The first three values in a orientation field is the axis of rotation. In this project, only one axis of rotation was used, the positive Y axis, upon which the camera rolls, thus reading 0 1 0. The fourth value in a orientation field is the amount of orientation this is where a radian value is entered. A full 360 degree turn can be calculated in radians as 2π. Thus 180 degrees would 3.142(π), 90 degrees would be 1.57 and 270 degrees would be 4.71 radians as illustrated in Figure 23.

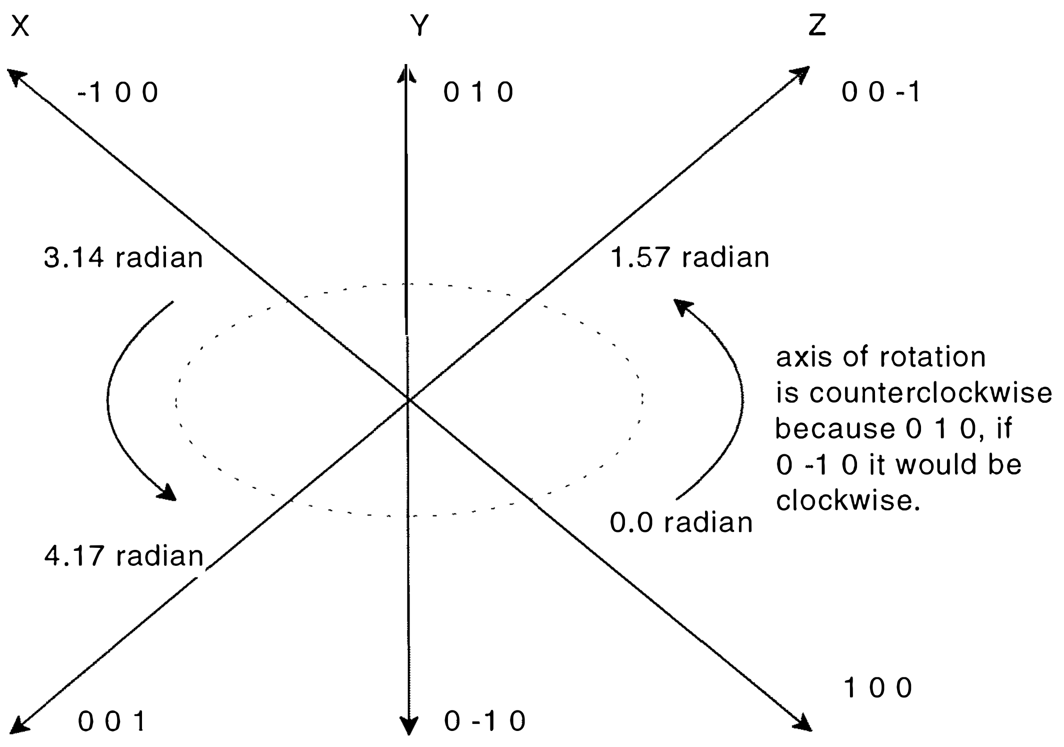

X
Y
Z
-1 0 0
0 1 0
0 0 -1
3.14 radian
1.57 radian
axis of rotation is counterclockwise because 0 1 0, if 0 -1 0 it would be clockwise.
0.0 radian
4.17 radian
0 0 1
0 -1 0
1 0 0

Figure 23: Orientation in VRML

The users current viewpoint orientation is 0 1 0 1.57. From a birdseye view of the campus a better understanding of how radians compare to the VRML world project can be seen in Figure 24.

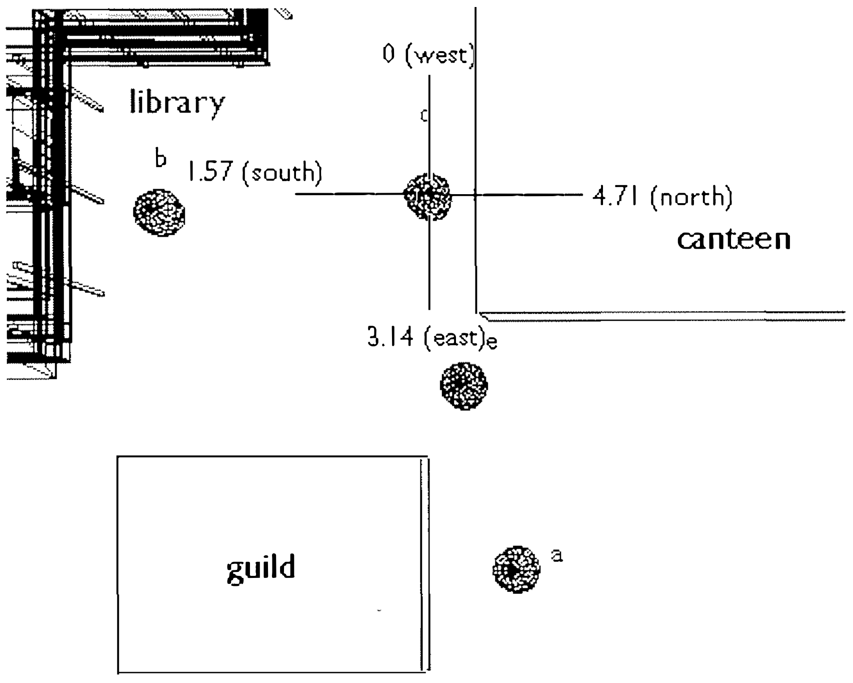


Figure 24: Orientation in VRML project

Knowing that the path is [d,e,f], the radian angle between d (canteen) and e (n1) is calculated first. The angle between the user's current position and the closest coordinate was not calculated because the whole animation works by the viewpoint moving to the nearest neighbour coordinate first, and the nearest neighbour coordinate would then hold the orientation from the user's original orientation (0 1 0 1.57). Then it is a matter of finding the orientation from the 'd' node to the 'e' node. This is achieved by first finding the sine of the angle, which is then converted into radians. This is illustrated below in Figure 25.

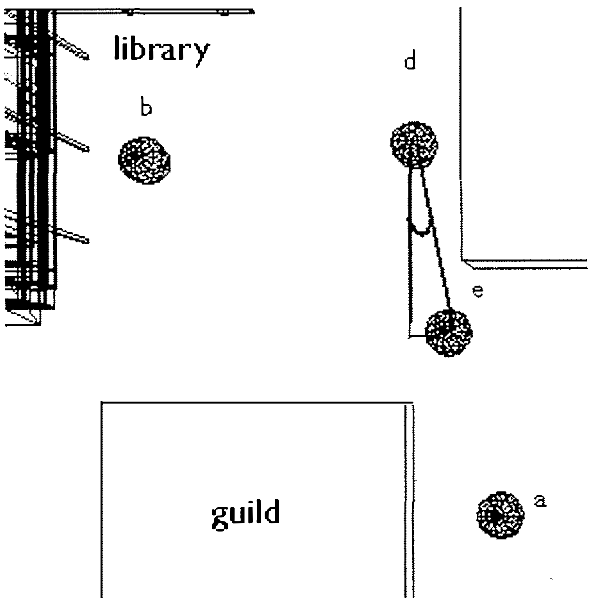


Figure 25: calculating the orientation

The sine of the angle is calculated using opposite/hypotenuse and is then converted into radians which in this case is [d,e] is .17 radian. To check what quadrant this angle is in, the difference between the X and Z values in the two coordinates is needed. As shown in Figure 26;

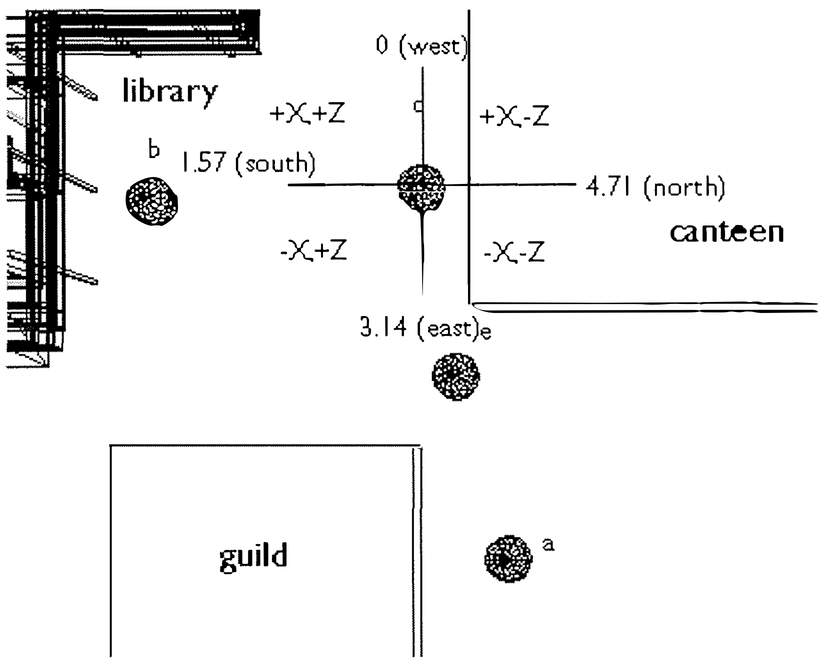


Figure 26: quadrant of the orientation

As the calculation below shows both X and Z have produced negative results thus the radian of 3.14 is added to the already existing angle of .17 making the angle of rotation 3.31.

canteen coordinate		279.1	1.6	-224.1
n1 coordinate	-	282.1	1.6	(-207.5)
		-3		-16.6

Then the angle for between n1 and building 6 is calculated which gives a rotation of 4.67 radian. The final orientation array would look like the following;

```
[[0, 1, 0, 1.571],        //users current orientation
[0, 1, 0, 3.3207],        //orientation to n1
[0, 1, 0, 4.67],          //orientation to building 6
[0, 1, 0, 0]]             //anomaly that is ignored
```

The final orientation in this example is 0, however because the animation stops before this at the last coordinate this orientation is never done because it is never sent to the VRML world.

Thus the two arrays that will be sent to VRML file read as such;

Coordinates

```
[[277.1, 1.6, -226.1], //current position
[279.1, 1.6, -224.1], //nearest neighbour
[282.1, 1.6, -207.5], //shortest path
[347.5, 1.6, -204.7]] //destination
```

Orientation

```
[[0, 1, 0, 1.571],       //users current orientation
[0, 1, 0, 3.3207],       //orientation to n1
[0, 1, 0, 4.67],         //orientation to building 6
[0, 1, 0, 0]]            //anomaly that is ignored
```

Presently nothing has been sent to the VRML world. At the end of converting the path into coordinates and orientations a function executes that sends the first data to show the user the shortest path.

Originally this is where the project might have ended. The two arrays could have been sent in whole to the PositionInterpolator and OrientationInterpolator, but this led to many problems. One problem was deciding on how large the array in the Interpolators was to be. This is because the applet creates a array whose size is determined by the size of the path. However the size of the Interpolators had to be fixed on initialisation of the VRML world.

Once the world is loaded the number of keyValues cannot change although its values can. Unfortunately finding the shortest path might produce the need to use either 5 sets of coordinates to complete the path or 10 or any number that is possible for the data structure to produce. To overcome this problem the array was set for 20 coordinates and 20 orientations which made the interpolators look like the following,

```
DEF CAMTIME  TimeSensor { enabled FALSE cycleInterval 0 },
DEF CAMPOS PositionInterpolator {
key[0,.05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,.75,.85,
.9,.95,1 ]
```

```
      keyValue [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
      0] },
      ROUTE CAMTIME.fraction_changed TO CAMPOS.set_fraction
      ROUTE CAMPOS.value_changed TO CAM.set_position

      DEF CAMROT OrientationInterpolator{
      key
      [0,.05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,.75,.85,.9,
      .95,1 ]
      keyValue [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] },
```

In order to fill out the rest of the keyValue field, the last set in the coordinates array was used to fill out the whole array in the interpolators. Such as, if their were 7 sets of coordinates once the seventh coordinate was placed in the position array the remaining 13 places in this keyValue field was filled out with the coordinate from the 7th position. However, in the final analysis this was not so much of a problem as the interpolation was turned off once it reached its goal. Also if this was not done the animation would simply sit on the spot for what time was remaining.

The choice of cutting out the interpolator all together and to use Java to construct interpolators which fed the Viewpoint directly, was experimented as another option. Unfortunately it proved unreliable, because the Java applet was now sending hundreds of new positions to the Viewpoint in a short amount of time through the EAI. This crippled the system because it was also monitoring the Viewpoint's eventOuts through the callback. Thus the idea was abandoned and the original large array in the PositionInterpolator was used again. The introduction of the orientation of the camera also led to more problems, that needed a new way of executing the path. This was because the orientation would occur over the duration of the viewpoint moving from one coordinate to another, producing a waltz like movement through the world.

The project's aim was that the orientation of the animated path was set to occur at every coordinate. To what angle the orientation occurs depends on the position of the next coordinate so that in the final animation the following occurs; move to position, stop moving, orientate to next position, move to position, stop moving, orientate to new position etc.

In order to do this however the interpolation under the original idea had to stop and start for each interpolator, i.e. one would execute then stop and then the other would execute and stop and then back to the other interpolator creating a ping pong effect.

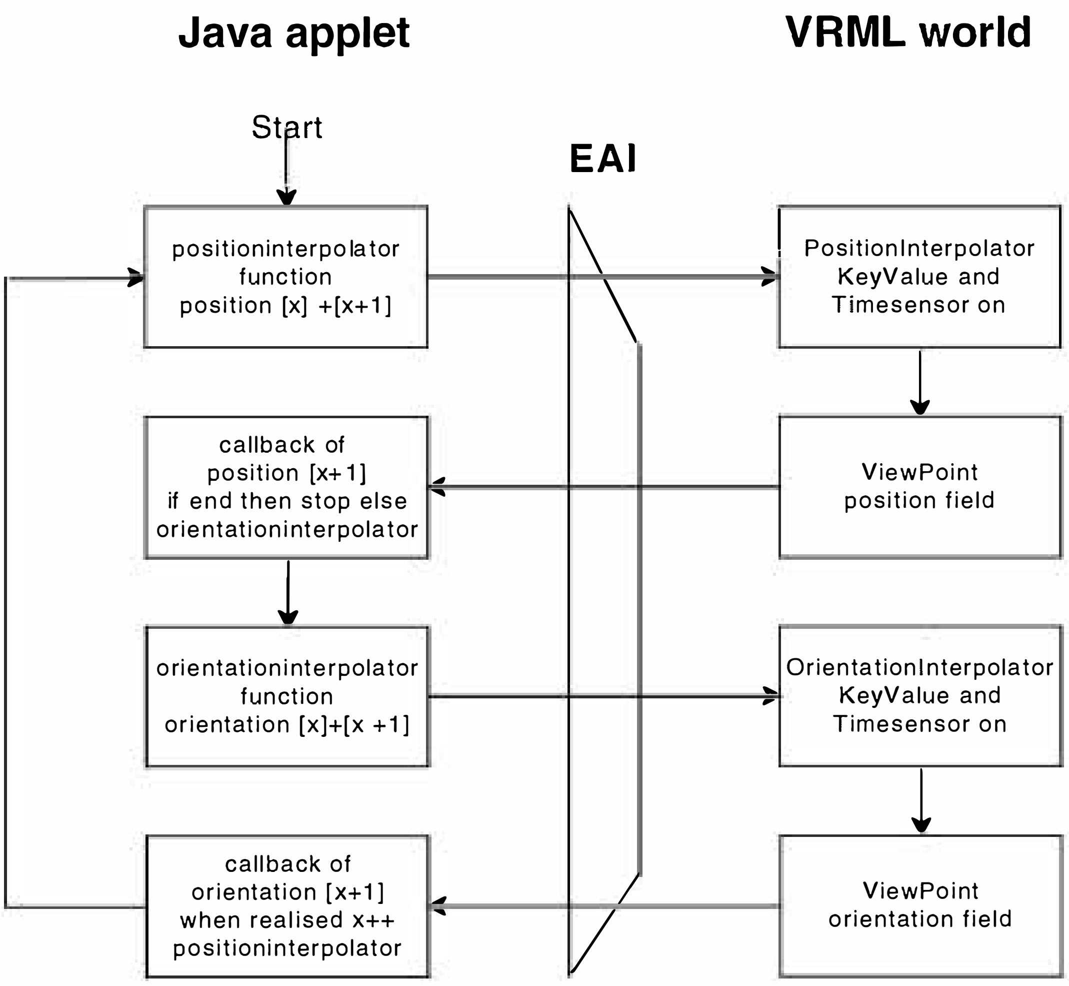
Originally this was to be done setting the TimeSensor to start generating its fraction_changed eventOut to at a certain position for example .5 for the middle of the animation etc. But this method proved clumsy and it did not solve the original problem of having a large array for interpolators which had to cover the maximum number of edges in a path which is n-1 (n = number of nodes).

This is where two more functions were added to solve the problem, and the callback was used more extensively than just using it to get the last position to turn the animation off. Under the new set up, the interpolators could only accept 2 sets of coordinates and orientations, as shown below;

```
DEF CAMTIME  TimeSensor {enabled FALSE},
DEF CAMPOS PositionInterpolator {
            key [0,1]
            keyValue [0 0 0 0 0 0 ] },
ROUTE CAMTIME.fraction_changed TO CAMPOS.set_fraction
ROUTE CAMPOS.value_changed TO CAM.set_position

DEF OCAMTIME  TimeSensor {enabled FALSE},
DEF CAMROT OrientationInterpolator{
            key [0,1]
            keyValue [0 0 0 0 0 0 0 0 ] },
ROUTE OCAMTIME.fraction_changed TO CAMROT.set_fraction
ROUTE CAMROT.value_changed TO CAM.set_orientation
```

Thus the whole system works in the following diagrammatic way. More detail of this is given in the proceeding pages.


Java applet
VRML world
Start
EAI
positioninterpolator function position [x] +[x+1]
PositionInterpolator KeyValue and Timesensor on
callback of position [x+1] if end then stop else orientationinterpolator
ViewPoint position field
orientationinterpolator function orientation [x]+[x +1]
OrientationInterpolator KeyValue and Timesensor on
callback of orientation [x+1] when realised x++ positioninterpolator
ViewPoint orientation field

Once the previous function converted the path into the two arrays the first part of the animation was executed. The pseudocode for this function is shown below.

```
positioninterpolator
IF x <= number_of_characters_in_shortest_path -1
    SET PostionInterpolator_keyvalue TO position_array [x] and position_array[x+1]
    SEND PostionInterpolator_keyvalue TO WORLD
    SET animation in PostionInterpolator  WORLD on
ENDIF
```

(Java source code pg. 135)

Once this is executed the first two coordinates from the position array are sent to the PositionInterpolator's keyValue field.

```
[[277.1, 1.6, -226.1], //current position
[279.1, 1.6, -224.1]] //nearest neighbour
```

The Timsensors controls are also set up and turned on to start the animation. The Viewpoint then starts to animate from the users present position to the canteen coordinate, the nearest neighbour.

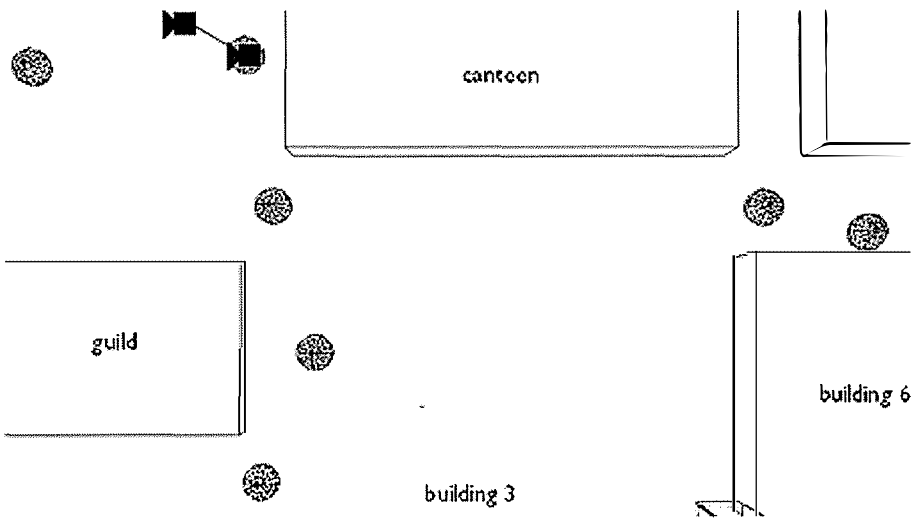


Figure 27: Viewpoint moves from original position to canteen

Once these coordinates were sent to the PositionInterpolator, the callback (EventOutObserver) would receive the coordinates of the Viewpoint and waited until the appropriate coordinate in the position array[x+1] had been met, as the following pseudocode illustrates;

```
callback_position
IF end_coordinate == Viewpoint_positon
        turn_all_animation_off
ENDIF
IF position_array[x+1] == Viewpoint_position
        turn_PositionInterpolator_animation_off
        orientationinterpolator
ENDIF
```

(Java source code pg. 124)

The callback for the Viewpoint's position performs checks for two values. The first thing it does is to see whether the Viewpoint is at the final position, if it is, everything is turned off and the animation stops. If it is not, then it checks as to whether the last coordinate in the first part of the path has been sent to the Viewpoint's position field. Once it has then the TimeSensor is turned off for the PositionInterpolator (thus turning off the animation, momentarily) and the orientationinterpolator function is executed, which is shown below in pseudocode.

```
orientationinterpolator
IF x <= number_of_characters_in_shortest_path -1
        SET OrientationInterpolator_keyvalue TO orientation_array [x] and
            orientation_array[x+1]
        SEND OrientationInterpolator_keyvalue TO WORLD
        SET animation in OrientationInterpolator WORLD on
ENDIF
```

(Java source code pg. 134)

When this is executed, the first two orientations from the orientation array are sent to the OrientationInterpolator's keyValue field as shown;

```
[[0, 1, 0, 1.571],        //users current orientation
[0, 1, 0, 3.3207]]        //orientation to n1
```

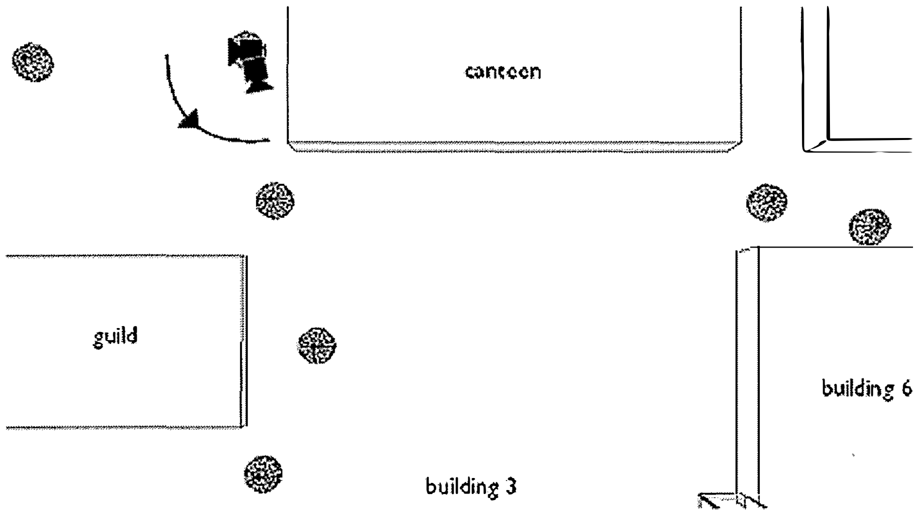


Figure 28: Viewpoint orientates towards the n1 node 'e'

So the first orientation that the user sees is the Viewpoint orientating from its original position to face the new coordinate that it is heading too, in this case to n1.

Another callback then monitors the Viewpoint's orientation field to check that the last orientation has been sent from the OrientationInterpolator to the Viewpoint's orientation. The following pseudocode traces this callback;

```
    callback_orientation
    IF orientation_array[x+1] == Viewpoint_orientation
            turn_OrientationInterpolator_animation_off
            x++
            positioninterpolator
ENDIF
```

(Java source code pg. 125)

Once the orientation has been realized in the Viewpoint, then the OrientationInterpolator is turned off and the index is incremented by one (x++) and the positioninterpolator function is executed again thus moving down one row in the position array and sending;

```
positioninterpolator
[[279.1, 1.6, -224.1], //canteen
 [282.1, 1.6, -207.5]] //n1
```

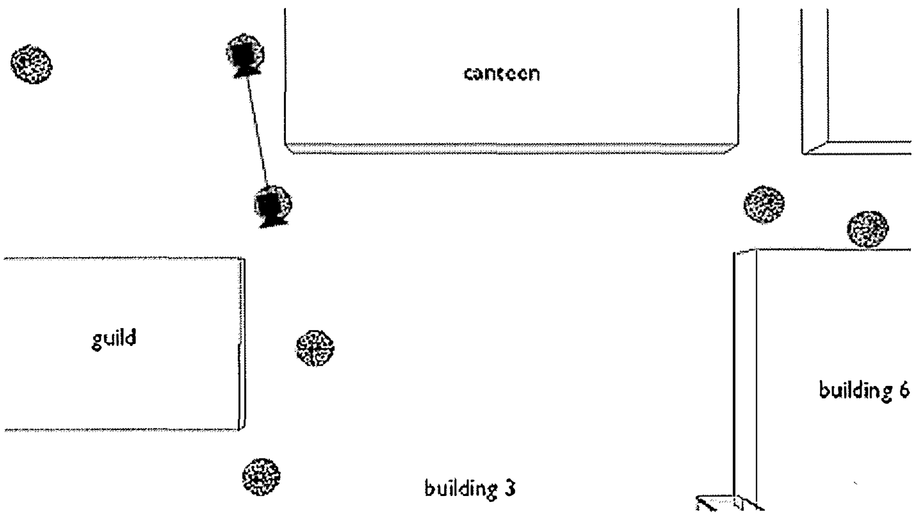


Figure 29: Move from canteen to n1 node

and so on;

```
position_callback
orientationinterpolator
[[0, 1, 0, 3.3207],      //orientation to n1
 [0, 1, 0, 4.67]]        // orientation to building 6
```

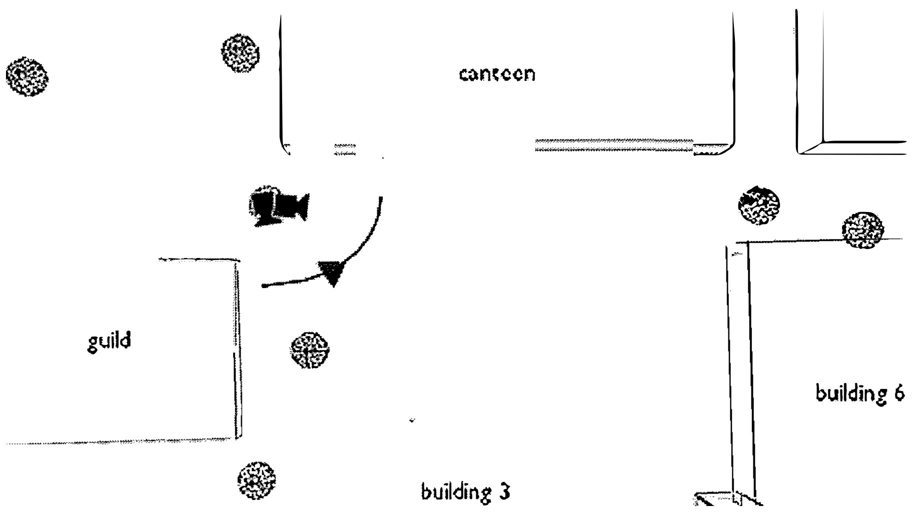


Figure 30: orientate towards building 6 node

```
orientation_callback
positioninterpolator

[[282.1, 1.6, -207.5], //n1
 [347.5, 1.6, -204.7]] //destination (building 6)
```

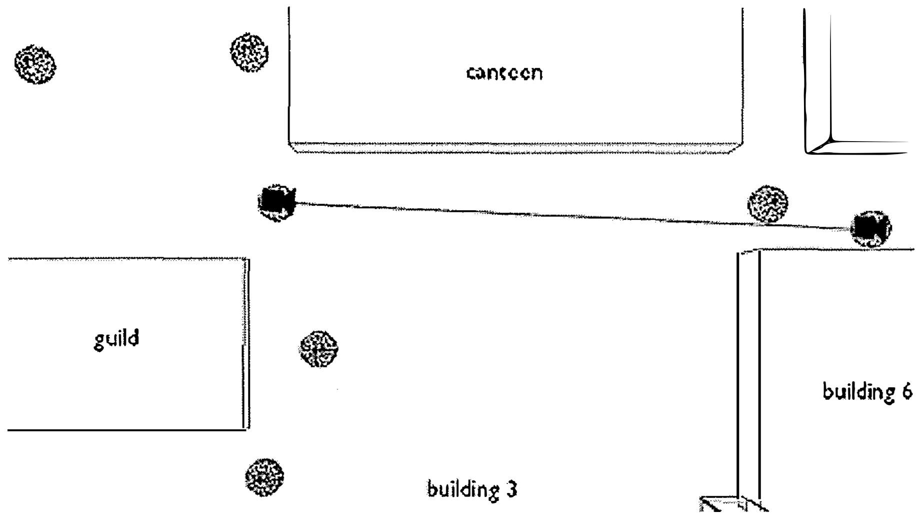
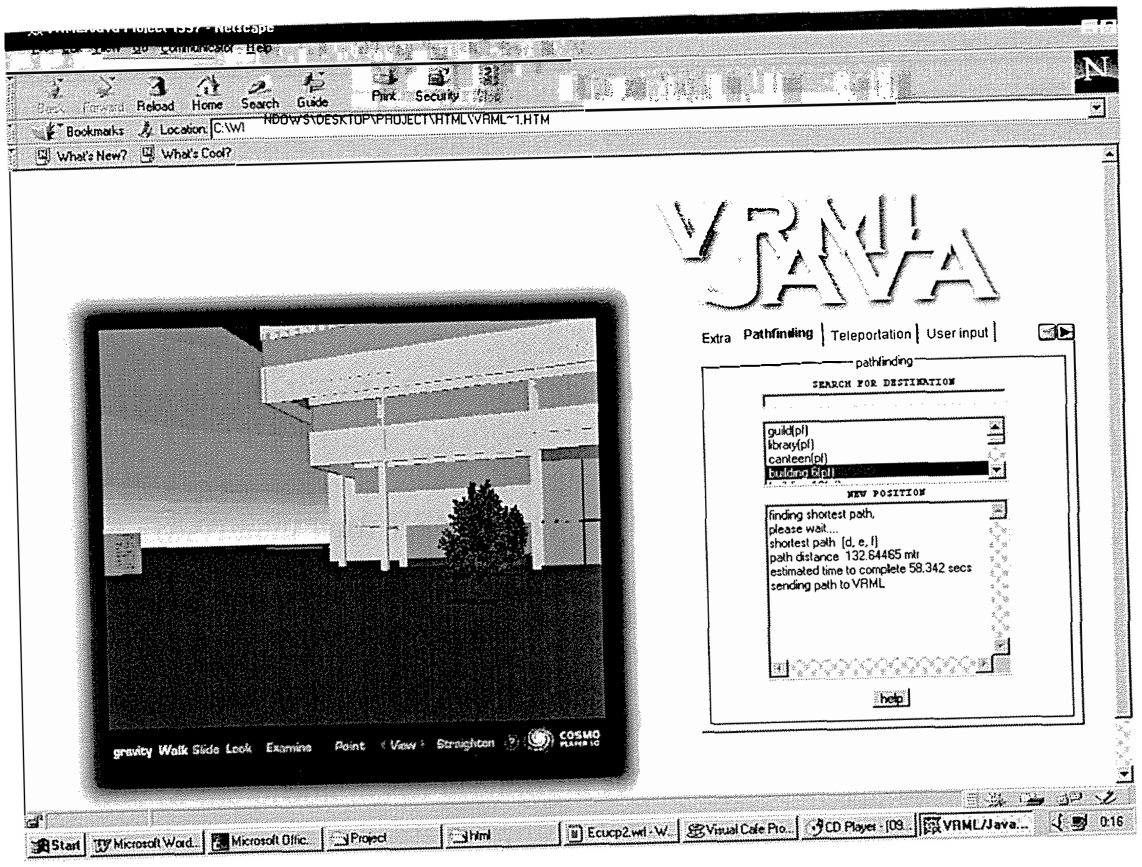


Figure 31: Move to building 6 node

```
all animation turned off.
```

This whole process goes from the positioninterpolator - callback - orientationinterpolator - callback - positioninterpolator etc. until the Viewpoint arrives at its coordinate destination upon which the animation is turned off and the path is complete. Figure 32 is a screen shot of the first orientation of this animation. The Viewpoint starts to rotate from the canteen coordinate towards the n1 coordinate, in a counterclockwise direction.



```
finding shortest path,
please wait....
shortest path [d, e, f]
path distance 132.64465 mtr
estimated time to complete 58.342 secs
sending path to VRML
```

Figure 32: first orientation in VRML as viewpoint begins to roll from canteen to face n1 node

Results

In order to appraise the results of this project the original goals of the project will be discussed in conjunction with what the final project delivers.

The original goal of the project was to,

"To experiment with VRML/Java using pathfinding as an example"

The result being that VRML alone at the present time, although having the potential, could not fully meet this requirement. However this was due to the problems of creating a HUD that could be used as an interface, a situation that could be resolved with future releases of the VRML specification.

As a tool, VRML and Java together can form a powerful relationship. Although separately both could deliver a solution (Java can be used to display 3D graphics) it is together that the two can deliver a better solution.

The original aim of the project was to investigate the necessities for pathfinding in a 3D space which has the 6 degrees of freedom X, Y, Z, roll, pitch and yaw. Four of these freedoms are inherent within the pathfinding project. The other two, pitch and yaw, were not considered for two reasons

- The Viewpoint node was used to simulate human movement as such humans do not pitch or yaw their head when walking. Originally the camera was to pitch when it went up the stairs to the first floor of the library but it seemed an unnatural movement, instead the camera faces forward and moves up the stairs.

- Pitch and yaw are simple extensions to the roll of a camera. In the case of the roll of the camera the angle was calculated on the X Z plane for pitch it would need to be on the X Y plane for yaw Y Z plane. The orientation would need to pivot around on the X axis and Z axis respectively. Roll in this project uses the Y axis.

The A* algorithm performed very well. After slightly modifying it a number of times, it proved to be extremely accurate even after extensive testing with over 40 different coordinates distributed around the X Y Z coordinate space. It must be stressed though, that the data structure determines the best algorithm, in this project it happened to be the A* algorithm. For other projects, with different data structures in that nodes are closely bound together etc the results would be different, hence the use of caution in choosing an algorithm to fit the data structure.

The ability of the A* algorithm to be 100% accurate, suggests that even though it is computationally efficient perhaps further refinement of the algorithm could have been achieved without losing accuracy (it took approximately 9 seconds to find the shortest path from one side of the campus to the other on different levels with over 40 coordinates in the graph).

VRML itself, after it was set up to accept small pieces of the path, performed the animation well at a high frame rate making it for smooth animation. There was however a trade off for the smoothness of the animation, and that was the speed of the animation.

The animation occurred at approximately 1mtr/sec. VRML however, is able to cope at a much faster speed than this and still retain smoothness in the animation. The problem was the callback from the VRML world to the Java applet. As each new position or orientation animation occurred, the number of new positions or orientations sent to the Viewpoint node is determined by the size of the cycleinterval field (which determines the length of the animation) in the TimeSensor node. If the value was low, then the interpolator would send new events to the Viewpoint at a much faster speed than if the cycleinterval was higher.

The problem seemed to be that the EAI was not sending back all the events, especially the last event which was critical for the next part of the animation to occur. This is why in the Java code for the project, the callback looks for the integer value of the position and not the floating point value. This means that when the animation has stopped at the destination the actual final position is within 1m^2 of its accurate final position which is quite acceptable. For the orientation though, a range had to be used even when the animation was at 1mtr/sec this range was 15 degrees .3 radians in either direction any lower and the callback might miss the needed orientation of the last value. These problems though relate to the slowness of the hardware and could not be overcome.

Another problem that was never completed for the orientation of the camera, more particularly the roll, was the problem of the camera rotating only in one direction (counterclockwise). It could be possible to almost do a 360° turn when facing the next node because the axis of rotation should have been 0 -1 0 (clockwise) instead of remaining at 0 1 0. At the time of writing though it was realised that this could have been solved by finding out if the angle of difference was greater than 180° then the axis of rotation needed to be at 0 -1 0 if lower then 0 1 0.

However, the results of this project has resulted in a number of key steps that need to be performed in order to complete pathfinding within a VRML world, Figure: 33 indicates the procedure to achieve this.

```
START
1.    Get destination (end node).
2.    Find users current position.
      2.1 Map users current position to the closest coordinate
          in data structure (root node).
3.    Find shortest path between start node and end node.
4.    Convert shortest path into VRML coordinates.
      4.1 Calculate orientation from VRML coordinates (pitch
          and yaw would be done here too).
5.    REPEAT until end node is realised.
      5.1 Send coordinates[x] & coordinates[x+1] to VRML
          world.
      5.2 Callback of coordinates, IF [x+1] = Viewpoint
          position.
      5.3 Send orientation[x] & orientation [x+1] to VRML
          world.
      5.4 Callback of orientation, when [x+1] = Viewpoint
          orientation.
END
```

Figure 33: Procedure for pathfinding in VRML/Java

Concluding this chapter, the results of the project were very encouraging. VRML has enough flexibility, with the help of Java, to perform pathfinding at a high level. As the hardware problem encountered in the project becomes redundant due to faster hardware, Java/VRML could be used to calculate and animate shortest path algorithms very accurately.

Conclusion

This chapter will point to further research that could be conducted into the field of VRML and its possible use in the future. A conclusion on the study will finish the thesis.

Further research

Finding shortest paths is one of the basics of artificial intelligence and this project shows that VRML has the necessities to have built in basic artificial intelligence. Coupled with the fact that VRML has the use of the WWW as its distributive medium, it demonstrates that the benefits of 3D graphics are no longer constrained to the realm of specialised computers.

With 3D graphics becoming a rapidly expanding area due to falling prices (hardware in particular), there exists an enormous amount of research to be done in this area. In particular with VRML 2.0 having been released a little over a year ago, there exists a huge gap of knowledge that is not known about the potential of VRML. Some further research into VRML is shown below.

- To expand this project by making pathfinding truly efficient, would require the use of a database that stored all objects in the VRML world. This would give the Viewpoint the ability of spatial reasoning. Human beings do not use set coordinates to find paths, we instead use our ability to think in 3D and spacialise our surroundings and store objects in our own database, this ideally is the goal in pathfinding. However this would require extensive work to be done in the area of artificial intelligence.

- HUD development for VRML - the development of a HUD Node in VRML was under discussion for VRML 98, although now it does not appear to be a part of the specification. The ability for a user to choose from a list, for a HUD to change options according to the user's decisions etc are all issues regarding the building of HUDs for virtual environments. The issue of creating 3D user interfaces has been an area of research for a number of years. It is thought that the next generation of operating systems would be in 3D.
- Smart bot for VRML - the use of a Java applet that has the ability to collect data all over the WWW and illustrate this data in VRML has enormous potential, because 3D has the ability to show relationships between data that sometimes cannot be seen on a 2D level.
- VRML integrity - VRML worlds can be downloaded from any server in the world. When one user downloads a copy of a VRML world, enters the world and opens a closed door, how does a user who downloads the same world get notification that the door is now open and not closed as it was previously. This is a problem of retaining the integrity of the world no matter how high the number of users. On a local network scale, this is not a problem. On the global network of the WWW, this is a problem. Multiuser worlds are common in VRML, Multiuser worlds and VRML integrity together, is not. There is also the problem of when entering a virtual world everything is in its initial state. If a user was to move an object, then leave the world and come back again by downloading it, the object would be back in its original position.
- Seamless environments - The ECU world created is a small self contained world. If a user was to catch a bus into the city of Perth which was contained in another world file and initialised once the user

entered the bus, then this new world would take time to load. But to get a truly virtual universe, seamless environments are needed. This feature would allow the user to go from one VRML world to another by walking down a road etc. To the user it would seem as though they were moving through one continuous virtual universe, even though their travels might take them to VRML content on dozens of machines around the real world.

- Environmental control in VRML - In the VRML world used in this project, a sun rotates around the ECU campus with its starting position based on the current system time. An avenue for more research exists on how to control other aspects of the VRML world through environmental factors i.e. seasons, weather (making rain in virtual reality) etc.

These are just a few of the possibilities of the future research in VRML, where this research will eventually concentrate is something only time will tell.

Conclusion

This project was used as a testing ground to investigate the potential of VRML in pathfinding and the control of VRML via the use of Java. The projects intentions proved to be quite successful.

As the field of virtual reality continually expands into new domains, there exists a large capacity for society to benefit from the realm of digital reality. With VRML being a standard format for 3D there is now a consensus that 3D graphics will explode in use.

There are several other 3D file formats that can do a better job than VRML but these formats are not open formats and suffer from the problem of accessibility to the technology. VRML, with the weight of the IT industry behind it however, will catch up fast as it took just 15 months to go from VRML 2.0 to VRML 98, future releases will probably be even shorter. Combined with the fact that VRML supports multimedia extensions such as streaming video, Java etc and as this project has done poses the ability to have at least basic artificial intelligence in pathfinding, VRML is set to become the benchmark for the creation of virtual environments.

There is no doubt that the future of computing will be 3D. Although the thought of using a word processing package with a 3D user interface seems ludicrous, it is because its different and has not been done yet.

"Virtual Reality won't merely replace Television, it will eat it alive!"

Arthur C. Clarke (Rheingold, 1991).

Definition of terms

Applet - A Java program that runs in the context of a Java - capable browser or the appletviewer. Java applets extend the content of web pages beyond just graphics and text. (Newman,1996).

Browser - A program used for reading, displaying and interacting with objects on the WWW.(Newman,1996).

class - a collection of variables and methods that an object can have, or a template for building objects.(Newman,1996).

a .class file - a file containing machine - independent Java bytecodes. The Java compiler generates .class files from source .java files for the Java interpreter to read. (Newman,1996).

Cross Platform - when an application works on several operating systems. Java works on both Macs, Windows, and Unix, so it is Cross Platform. (http://vrml.sgi.com/basics/)

HTML (HyperText Markup Language) - the scripting language with which Web pages are written. (Marrin & Campbell,1997).

Interpolators - Interpolators are built-in behaviour mechanisms that generate output based on linear interpolation of a number of key data points. (http://vrml.sgi.com/basics/)

Java - An object orientated programming language that can be used to create machine independent applications and applets. (Newman,1996).

Node (VRML) - A node in VRML implements some functionality, the name of the node indicates its functionality ie Transform, Cone etc. Each node contains a list of fields that define parameters for its function ie translation, rotate, bottomRadius. (Marrin & Campbell,1997).

Sensors - Provide mechanisms for the user to interact with objects in the world. (http://vrml.sgi.com/basics/)

Virtual Environment - A world that is simulated entirely within the memory of a computer. A virtual environment might consist of a three dimensional house, or visualisation of a set of complex data, or any number of things. It is through virtual reality that users are able to create and explore these virtual environments. (Aukstakalnis et al.,1992,p.12).

Virtual Reality - Jaron Lanier, ex-president of VPL, coined the term virtual reality as "a computer generated, interactive, three dimensional environment in which a person is immersed". (Aukstakalnis et al., 1992, p.12).

VRML (Virtual Reality Modeling Language) - A language used to create three-dimensional content on a Web site. It is also a file format for saving three-dimensional models. (Marrin & Campbell,1997).

World Wide Web - The servers and connections between servers that contain and deliver the information shared over the Internet. (Marrin & Campbell,1997).

References

Ames, A.Nadeau, D.Moreland, J.(1997). *VRML 2.0 Sourcebook.* New York: John Wiley & Sons.

Anon.(1996). Drew University offers a CD-ROM tour of its campus. *Link-Up. 30* (3), 18.

Aukstakalnis, S.(1992).*Silicon Mirage: The Art and Science of Virtual Reality*.Berkeley: Peachpit Press.

Bergsman, S.(1997).Virtual Reality.*Journal of Property Management. 62* (1) 26-29.

Brutzman, D.*"A virtual world for an autonomous underwater vehicle"*. [on-line]. Available WWW: http://www.stl.nps.navy.mil/~brutzman /dissertation.[1997, December 12].

Brown, David. *"Descent" navigation demo using the EAI.*[on-line]. Available WWW:http://vrml.sgi.com/developer/eai/index.html.[1997, December 10].

CimCentre.*"Pathfinder"*.[on-line]. Available WWW: http://cimcentre. snu.ac.kr/~next/viewer/pathfinder.html.[1997, December 11].

CASA.*'The Map of the Future"*.[on-line].Available WWW: http://www.bath.ac.uk/Centres/CASA/london. [1997, December 11].

Coolware.*"Intelligent Agents and Artificial Intelligence"*.[on-line]. Available WWW:http://www.coolware.com/lotech/3technology/4ai/. [1997, December 12].

Dvorak, P.(1997).Engineering puts virtual reality to work. *Machine Design. 69* (4) 69-73.

Grogono, P.Nelson, S.(1982).*Problem Solving and Computer Programming.* Massachusetts: Addison-Wesley.

Hamit, F.(1993).*Virtual Reality and the Exploration of Cyberspace.* Indianapolis: Sams Publishing.

Jones, M.Wyatt, A.(1994).*3D Madness.* Indianapolis: Sams Publishing.

Kevin, O.*"AI Lab Telerobotic Control page"*.[on-line] Available WWW: www.ai.mit.edu/projects/webot/robot/.[1997, December 12].

Leinfuss, E.(1996).Virtual worlds, real applications. *InfoWorld, 18* (48), 57-59.

Marrin, C.Campbell, B.(1997). *Teach Yourself VRML in 21 days.* Indianapolis: Sams.net Publishing.

Matzer, M.(1996).Fujitsu woos retailers for v-mall. *Brandweek, 37* (18), 16.

Marco, T.(1979).*Software Engineering.*New Jersey:Prentice Hall.

Newman, A.(1996).*Using Java.* Indianapolis: Que Corporation.

NTT.*"V.EXPO"*.[on-line].Available WWW: http://www.construct.net/ projects/ntt/.[1997, December 11].

Rheingold, E.Nievergelt, A.Deo P.(1980).*Combinational Algorithms.* Masachusetts: Addison-Wesley.

Ritchey, T.(1996).*Programming Javascript for Netscape 2.0.* Indianapolis:New Riders Publishing.

Sarna, D.Febish,G.(1996).The Business Reality of VRML. *Datamation, 42* (10), 27-30.

Schmidt, O.*No title*.[on-line]. Available WWW: http://www.rvs. unibielefeld.de/project/vrml-uni/).[1997, December 12].

Tilmann, R.*"A world of worlds"*.[on-line].Available WWW: http://www. meshmart.org/wow.[1997, December 11].

UCLA.*No title*.[on-line].Available WWW: http://www.gsaup.ucla. edu/vrml/).[1997,December 12].

Van der Linden, Peter(1996).*Just Java.*Mountain View,California: Sunsoft Press.

Vince, J.(1992). *3D Computer Animation.* Wokingham,UK: Addison-Wesley.

Vizard, M.(1996).It's time for VRML to grow up, earn its keep. *InfoWorld, 18* (32), 3.

VRML Architecture Group.*VRML Architecture Group.*[on-line] Available WWW: http://vag.vrml.org.[1997, Decemeber 10]

Watkins, C.Marenka, S.(1994).*Virtual Reality excursions.* Massachusetts: AP Professional.

Weiss,M.(1993).*Data Structures and Algorithms Analysis in Ada.* California: Benjamin/Cummings Publishing Co.

Wicks, John.*Finite Mathematics.*[on-line]. Available WWW http://www.northpark.edu/acad/math/courses /Math_1030/WebBook/ [1997, 10 December].

Winston, Patrick.(1984).*Artificial Intelligence.*Massachusetts: Addison-Wesley.

Appendix

The appendix is divided into the following five categories;

- Appendix A - HTML source code needed to display the applet and world file.
- Appendix B - Gallery of VRML world.
- Appendix C - VRML source code for the world file, because this file contains over 500 pages of text only the sections relating to the Java applet have been included, the code for the library and building 3 have not been shown.
- Appendix D - User guide for the Java applet.
- Appendix E - Java source code for the applet.

Appendix A HTML source code

```
<HTML>
<HEAD>
<TITLE>VRML/Java Project 1997</TITLE>
</HEAD>
<EMBED src="ecu.wrl"  HEIGHT=406 WIDTH=459 ALIGN=TEXTTOP>
<APPLET code="ecu.class" mayscript HEIGHT=400 WIDTH=380 ALIGN=TEXTTOP
></APPLET>
</BODY>
</HTML>
```

Appendix B Gallery of world file (ecu.wrl)

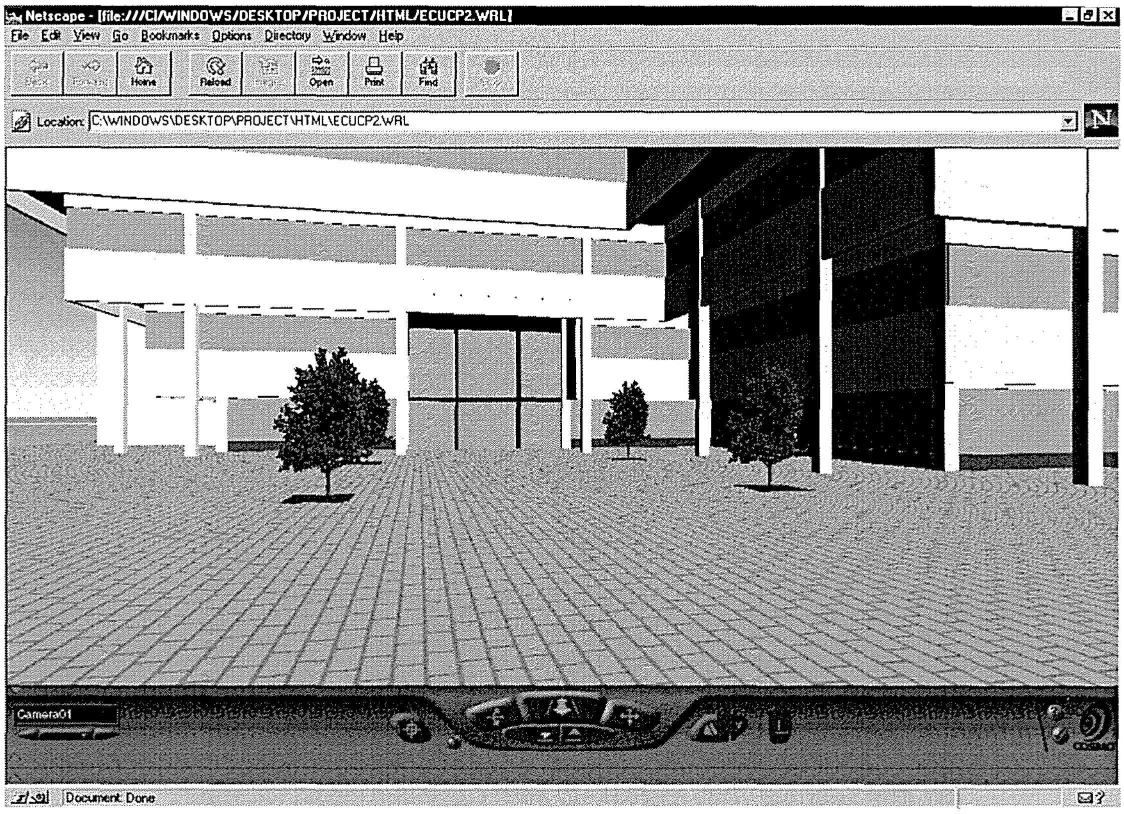

Netscape - [file:///C|/WINDOWS/DESKTOP/PROJECT/HTML/ECUCP2.WRL]
File Edit View Go Bookmarks Options Directory Window Help
Home
Reload
Open
Print
Find
Location: C:\WINDOWS\DESKTOP\PROJECT\HTML\ECUCP2.WRL
Camera01
Document: Done

Figure 34:ECU world outside the library in the afternoon

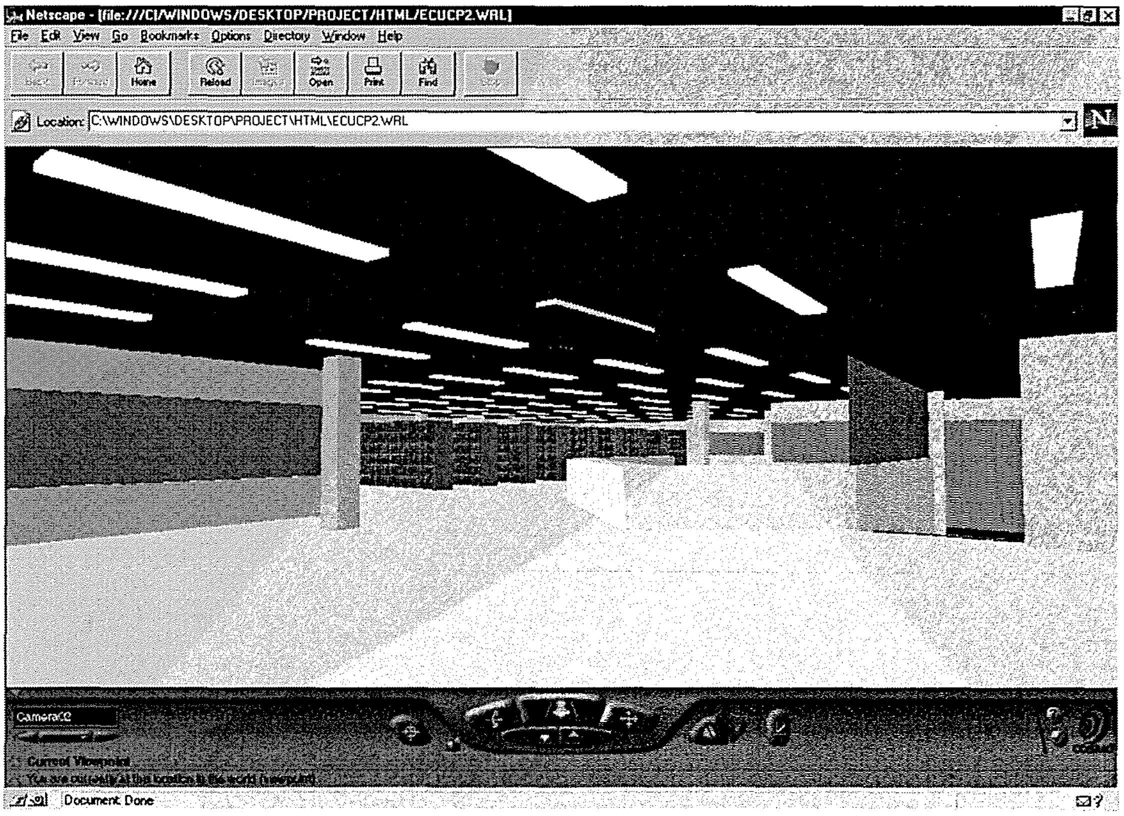

Netscape - [file:///C|/WINDOWS/DESKTOP/PROJECT/HTML/ECUCP2.WRL]
File Edit View Go Bookmarks Options Directory Window Help
Home
Reload
Open
Print
Find
Location: C:\WINDOWS\DESKTOP\PROJECT\HTML\ECUCP2.WRL
Camera02
Current Viewpoint
You are currently at this location in the world (viewpoint)
Document: Done

Figure 35: ECU world inside the library 1st floor

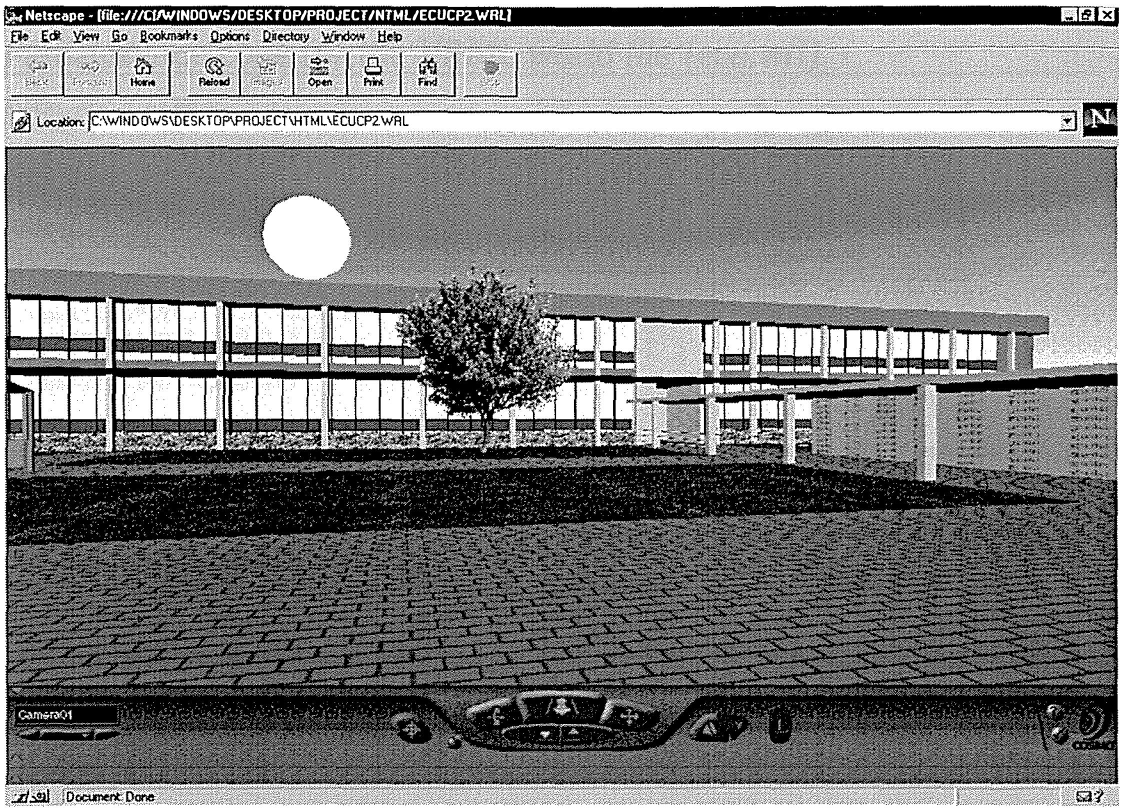

Netscape - [file:///C|/WINDOWS/DESKTOP/PROJECT/HTML/ECUCP2.WRL]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: C:\WINDOWS\DESKTOP\PROJECT\HTML\ECUCP2.WRL
Camera01
Document: Done

Figure 36: facing building 3 from the canteen early morning with sun

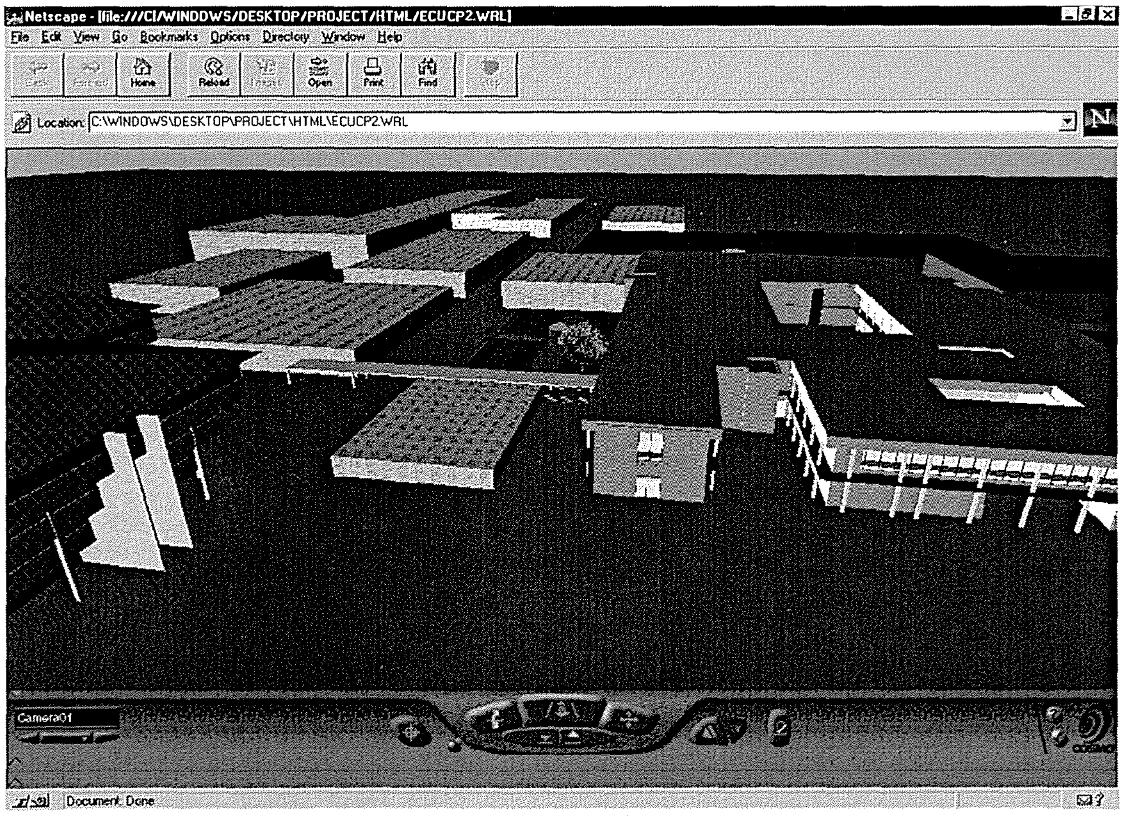

Netscape - [file:///C|/WINDOWS/DESKTOP/PROJECT/HTML/ECUCP2.WRL]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Location: C:\WINDOWS\DESKTOP\PROJECT\HTML\ECUCP2.WRL
Camera01
Document: Done

Figure 37: birdseye view of campus facing north, late evening.

Appendix C

Partial source code for the VRML world file (ecu.wrl)

```
#VRML V2.0 utf8


#the Viewpoint node used for the project, the intial viewpoint sits on the
#coordinate of the canteen facing in a southerly direction (towards the library).


DEF CAM Viewpoint {

  position 279.1 1.6 -224.1
  orientation 0 1 0 1.571
  fieldOfView 0.7854
  description "Camera01"
}

#The PositionInterpolator that was used to send new positions to the Viewpoint.

DEF CAMTIME  TimeSensor { enabled FALSE cycleInterval 7 },
DEF CAMPOS PositionInterpolator {
  key [0,1]
  keyValue [0 0 0 0 0 0 ] },
ROUTE CAMTIME.fraction_changed TO CAMPOS.set_fraction
ROUTE CAMPOS.value_changed TO CAM.set_position

#The OrientationInterpolator that was used to send new orientations to the Viewpoint.

DEF OCAMTIME  TimeSensor { enabled FALSE cycleInterval 5 },
DEF CAMROT OrientationInterpolator{
  key [0,1]
  keyValue [0 0 0 0 0 0 0 0 ] },

ROUTE OCAMTIME.fraction_changed TO CAMROT.set_fraction
ROUTE CAMROT.value_changed TO CAM.set_orientation

#Upon touching this building (guild) the TouchSensor would be activated
#(isOver = true) and the building is highlighted in the applet.

DEF oguild Transform {
  translation 266.4 1.2 -191.5
  children [
    Shape {
      appearance Appearance {
        material DEF MAT Material {}
          texture ImageTexture{url    "guildwrap.gif"}
            textureTransform TextureTransform {scale 10 10}
         }
      geometry DEF guild Box { size 26 2.4  18.45 }
    }
    ] }
        DEF TOUCH_GUILD TouchSensor {}
  ]
}

#the background colour of the world was changed according to the time
#of the day it was either black for night time or shades of blue for daylight
#hours

DEF SKY Background {
    skyColor [
        0.0 0.2 0.7,
        0.0 0.5 1.0,
        1.0 1.0 1.0
    ]
    skyAngle [ 1.309, 1.571 ]
    groundColor [
        0.1 0.10 0.0,
        0.4 0.25 0.2,
        0.6 0.60 0.6,
    ]
    groundAngle [ 1.309, 1.571 ]
}
```

```
#the sun rotated around the ECU world accroding to the time it indicated what
#the array was sent from the java applet, so that the sun would start at a different
#position relative to the time cyleinterval 86400 number of seconds in a day. This
#revolution takes two nodes around the campus a shape node (sphere) for the sun
#and a PointLight node for the sunlight

DEF Dummy01 Transform {
  translation 371.2 4.652 415.1
  rotation 0.5774 -0.5774 0.5774 -4.189
  scale 1 1 1
  scaleOrientation 0 0.4927 -0.8702 -0.2658
  children [
  DEF SUN_TS  TimeSensor { loop TRUE cycleInterval 86400 },
  DEF SUN_PI PositionInterpolator {
    key [0, 0.09, 0.18, 0.27, 0.36, 0.45, 0.54, 0.63, 0.72, 0.81, 0.9,
        1 ]
    keyValue [000 000 000 000 000 000 000 000 000 000 000 000] },
    DEF sunlight Transform {
      translation -13.54 -5.723 -63.57
      rotation 0 0 -1 -1.571
      scale 1 1 1
      scaleOrientation -0.4337 0.5024 -0.748 -0.9687
      children [
          DEF sunlight PointLight {
            intensity 2.5
            color 1 1 1
            location 0 0 0
            on TRUE
            radius 320

        }
    }
    DEF sun Transform {
      translation 2.26 -3.398 -60.13
      rotation 0.5774 -0.5774 -0.5774 -2.094
      scale 1 1 1
      scaleOrientation 0.03587 -0.1318 0.9906 -0.3952
      children [
        Shape {
          appearance Appearance {
            material Material {
              diffuseColor 1 1 .85
             emissiveColor 1 1 .85

            }
          geometry Sphere { radius 35 }

        }
      ]
    }
ROUTE suntime.fraction_changed TO sunpi.set_fraction
ROUTE sunpi.value_changed TO Dummy01.set_translation


#This proximity sensor was used to find the present position of the user as it
#encompassed the whole world file.


DEF PROXS2 Transform {
    children [
      DEF GLOBALPS ProximitySensor    {
        enabled TRUE
        size 1000 66 1000

        }
    ]
  }
```

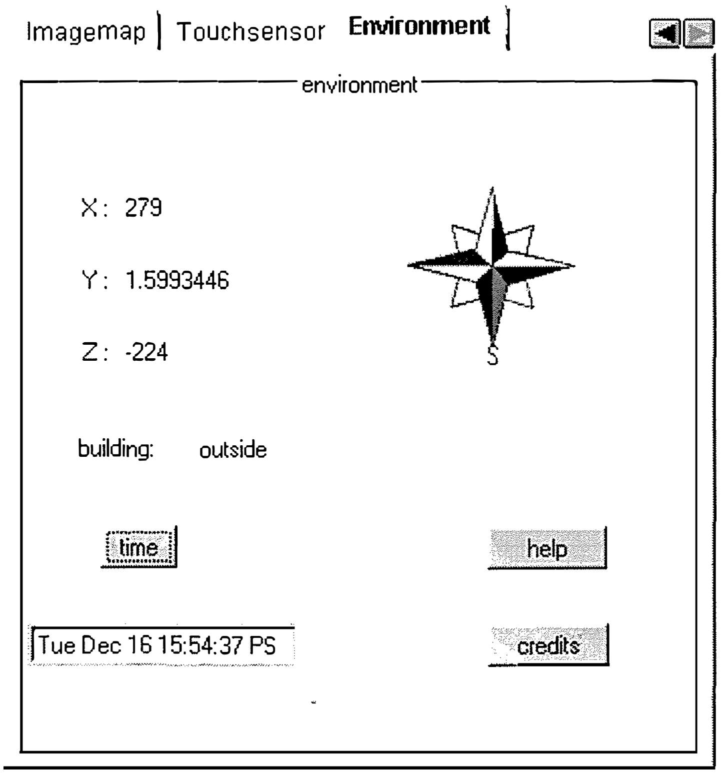
Appendix D User Guide for applet

This section describes the seven tab-panels used to build the applet and the subsequent functions available on each tab-panel.

Environment panel

This is a general panel, shown first, when the user enters the ECU world, and has the following general information about the user in the ECU world;

- Users current coordinate.
- What direction it is facing (north, south, west or east).
- Where they are located within the VRML world (building name or outside) .
- The current ECU time.
- It also has the facility for the user to enact the help frame and credits frame.



X: 279

Y: 1.5993446

Z: -224

building: outside

Tue Dec 16 15:54:37 PS

Figure 38: environment panel from the applet

Touchsensor panel

This panel provides the user with a 2D birdseye map of the campus and it highlights the building when the user clicks on the associated building within the VRML world, as well as giving the name of the building in the textbox. This is achieved through the use of a TouchSensor encompassing every building in the VRML world. It also has a small red dot just outside of the canteen when the user moves in the VRML world this red dot is moved to show the location of the user from a birdseye view. It is achieved by translating the Viewpoint position field into the position of the red dot on the Java applet.

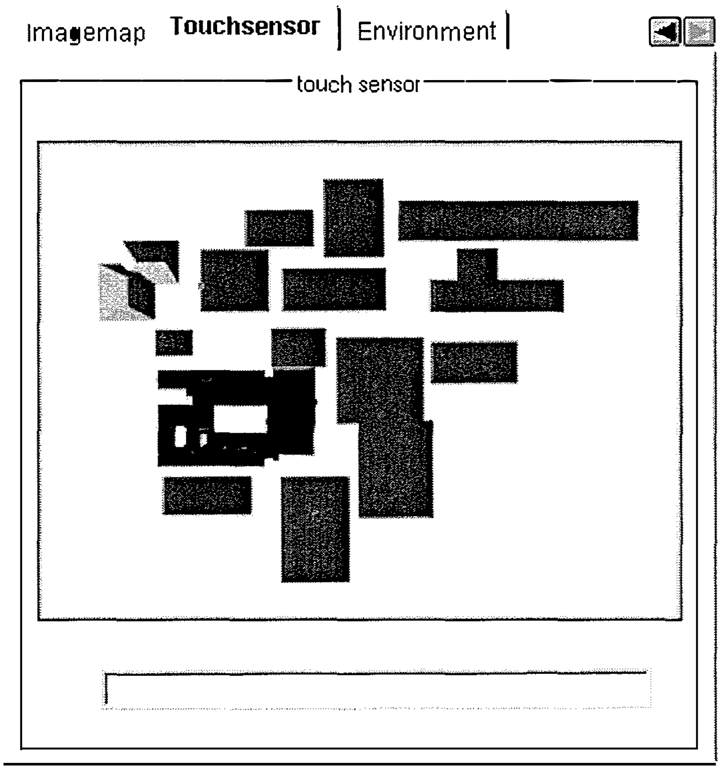


Figure 39: touchsensor panel from the applet

Imagemap panel

This panel acts as an imagemap. When the user rollsover a particular building in the applet, the name of the building is shown beside it. If a user clicks on the building, the user teleports to the entrance of that particular building in the VRML world. The coordinate of the user in the 3D world is also given and the moving red dot that relates to the last frame also appears here.

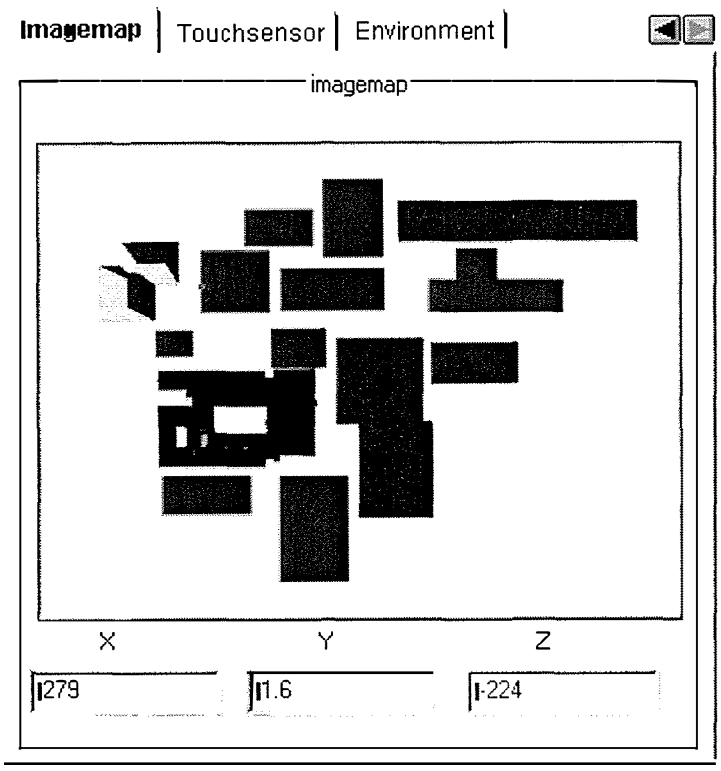


Figure 40: imagemap panel from the applet

Paths and Maps panel

Similar to the previous Imagemap panel but instead of teleporting when a user clicks on a particular building, the pathfinding algorithm is executed and the animated path between the user's current position and the end position is shown.

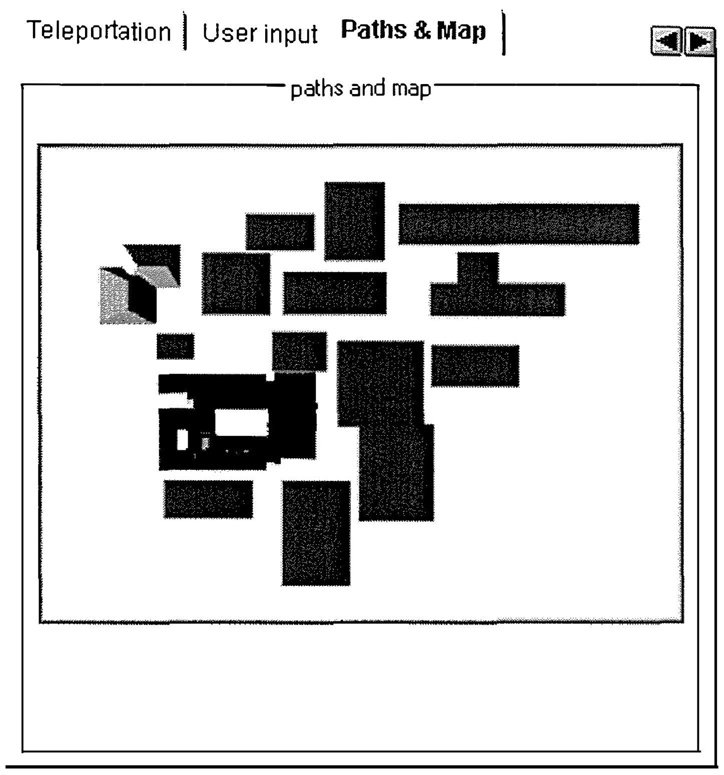


Figure 41: paths and map panel from the applet

User Input panel

This panel allows the user to input a particular coordinate and choose a direction. When the submit button is pushed, these new coordinates and direction are then sent to the Viewpoint within the VRML world.

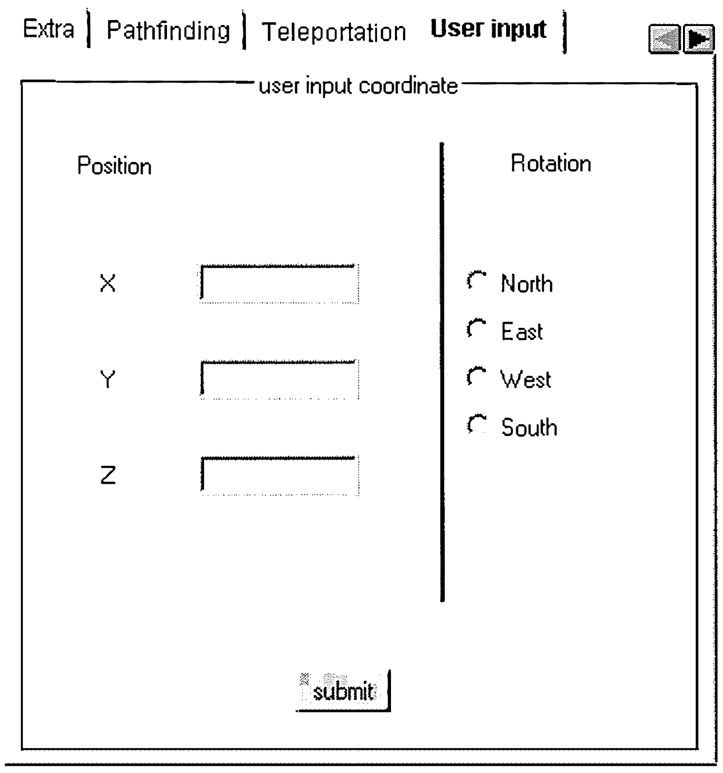


Figure 42: user input panel from the applet

Teleportation panel

This panel gives the user a list of options to choose from. When the user clicks on their choice from the list the user is teleported to the new position and is told of their new coordinate space and orientation within the VRML world. A search facility is also created for the user to quickly find a destination from the list.

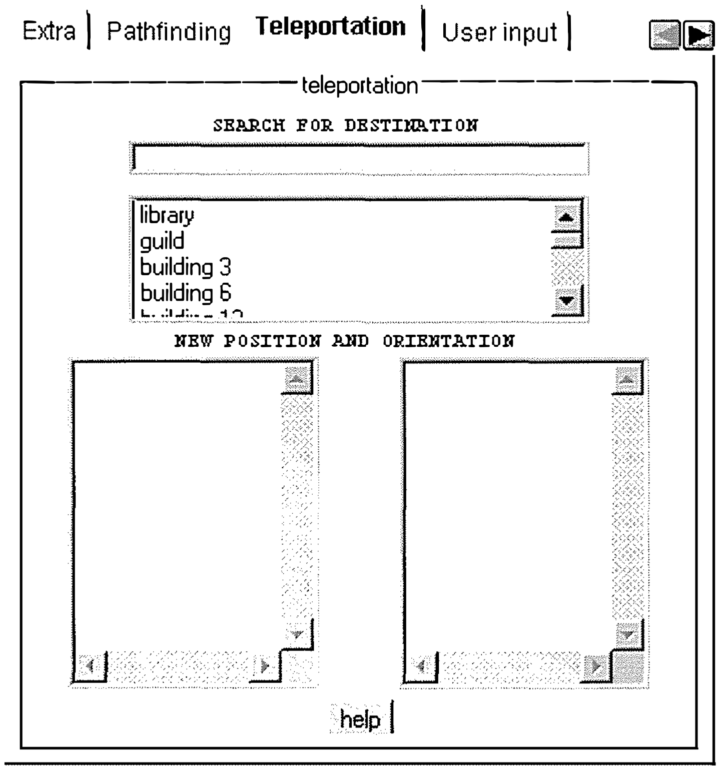


Figure 43: teleportation panel from the applet

Pathfinding panel

Similar to the teleportation panel, this panel however, finds the shortest path between the user and the destination and animates it through VRML's Viewpoint. Additional information such as the length of the path and the time it will take is shown in the text box below. Once the user has arrived the textbox tells the user they have arrived at the chosen destination.

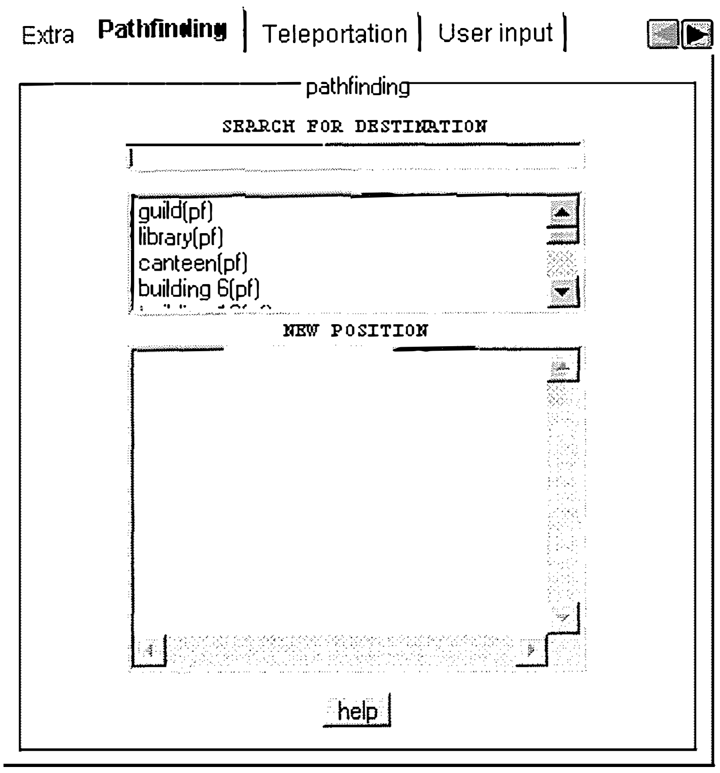
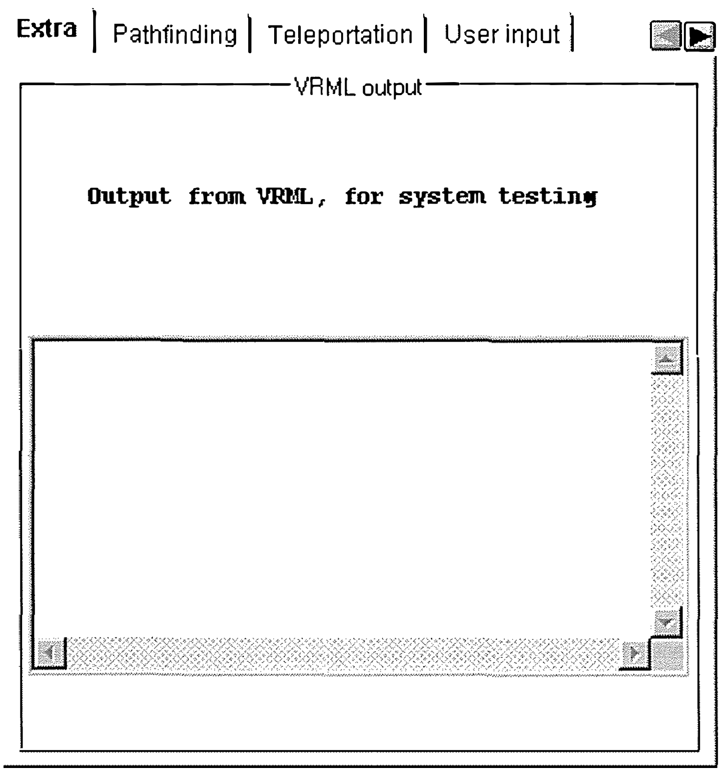


Figure 44: pathfinding panel from the applet

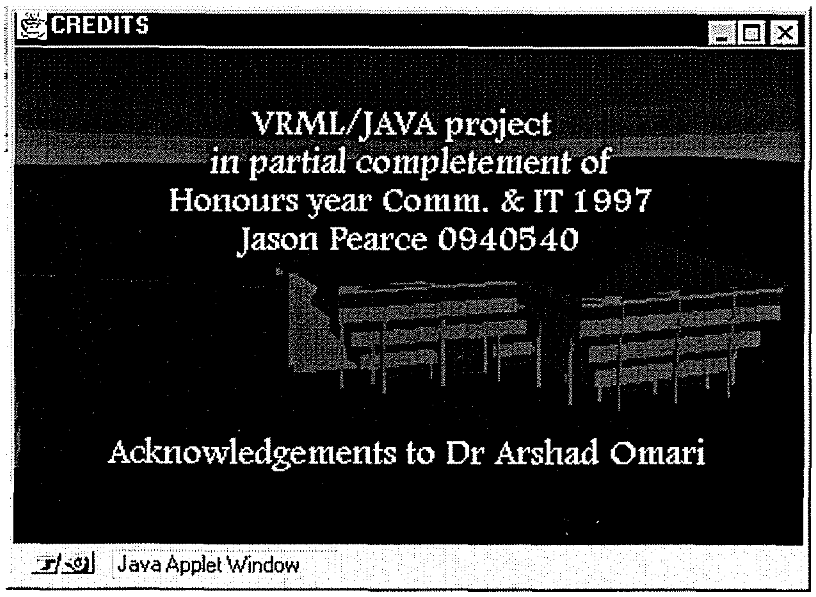
Extra panel - This panel was used for development testing, to output field values from the VRML file and Java variables.



Output from VRML, for system testing

Figure 45: extra panel from the applet

Credits frame



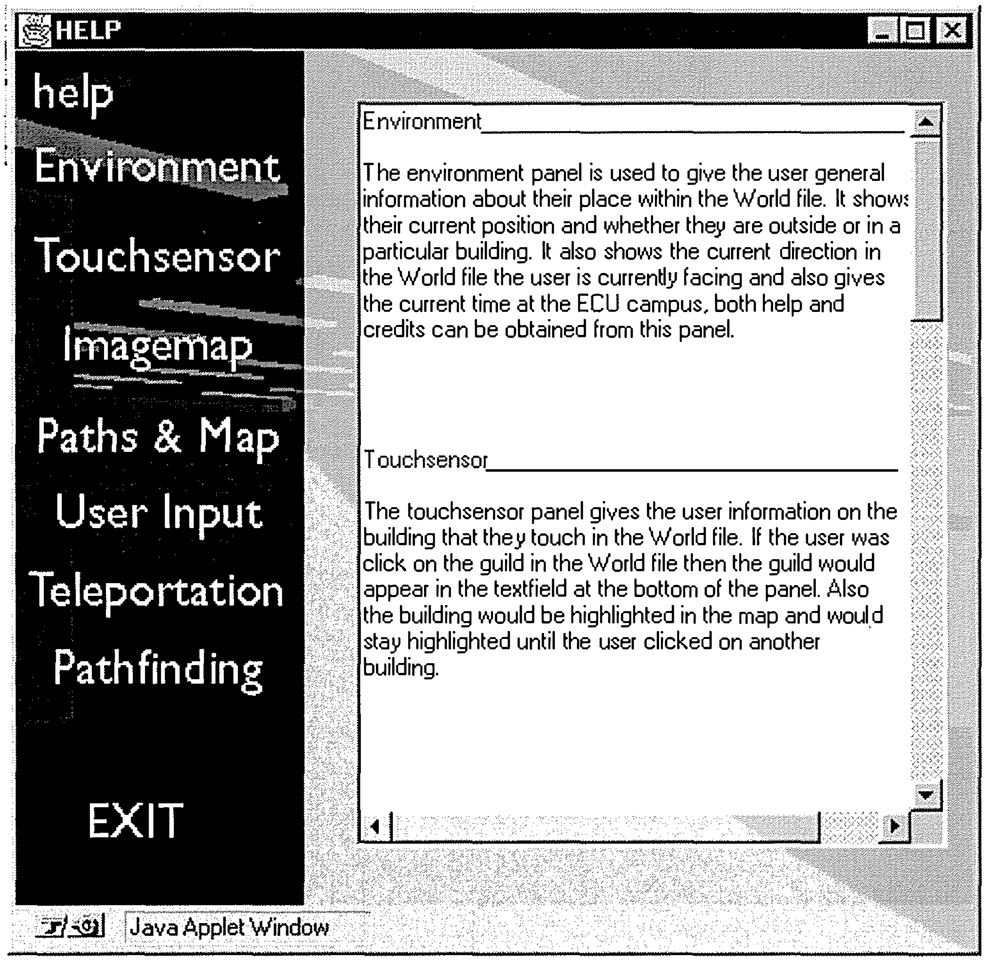
VRML/JAVA project
in partial completement of
Honours year Comm. & IT 1997
Jason Pearce 0940540

Acknowledgements to Dr Arshad Omari

Figure 46: credits frame from the applet

Help frame

Within the applet there is also a help frame to guide the user on the performance of the applet. As shown below;



Environment

The environment panel is used to give the user general information about their place within the World file. It shows their current position and whether they are outside or in a particular building. It also shows the current direction in the World file the user is currently facing and also gives the current time at the ECU campus, both help and credits can be obtained from this panel.

Touchsensor

The touchsensor panel gives the user information on the building that they touch in the World file. If the user was click on the guild in the World file then the guild would appear in the textfield at the bottom of the panel. Also the building would be highlighted in the map and would stay highlighted until the user clicked on another building.

Figure 47: help frame from the applet

Appendix E Source code for the Java applet

```
//ecu.java
/**************************************************
*****Java applet for the creation of shortest  *****
*****paths within the VRML world using the EAI.*****
*****Project completed in partial fulfillment  *****
*****of Comm & IT (Honours).                   *****
*****Jason Pearce 0940540                      *****
**************************************************/
import java.awt.*;
import java.applet.*;
import java.util.*;
import vrml.external.*;
import vrml.external.field.*;
import vrml.external.exception.*;
import netscape.javascript.JSObject;
import symantec.itools.awt.*;

public class ecu extends Applet implements EventOutObserver {

  //
  //VRML handles
  //
  Browser browser = null;

  EventInMFVec3f newsunpi = null;
  EventInSFBool newsuntime = null;

  EventInSFVec3f translation = null;
  EventOutSFVec3f newtrans = null;
  EventInSFRotation rotate = null;
  EventOutSFRotation newrot = null;

  EventInMFVec3f kv = null;
  EventOutMFVec3f newkv = null;
  EventInMFRotation okv = null;
  EventOutMFRotation onewkv = null;

  EventOutSFBool touchguild = null;
  EventOutSFBool touchcanteen = null;
  EventOutSFBool touchb14 = null;
  EventOutSFBool touchb15 = null;
  EventOutSFBool touchb16 = null;
  EventOutSFBool touchb17 = null;
  EventOutSFBool touchb13 = null;
  EventOutSFBool touchb6 = null;
  EventOutSFBool touchb18 = null;

  EventOutSFVec3f globalps = null;
  EventOutSFRotation oglobalps = null;

  EventInMFColor skychange = null;

  EventInSFTime timing = null;
  EventInSFTime stop = null;
  EventInSFTime start = null;
  EventInSFBool loopy = null;
  EventInSFBool enable = null;

  EventInSFTime otiming = null;
  EventInSFTime ostop = null;
  EventInSFTime ostart = null;
  EventInSFBool oloopy = null;
  EventInSFBool oenable = null;

  //
  //
  //data structure used for pathfinding.
  //
  //

  float south [] = {0f,1f,0f,1.571f};
  float west [] = {0f,1f,0f,0f};
  float north [] = {0f,1f,0f,4.712f};
  float east [] = {0f,1f,0f,3.142f};
float guild [] = {286.7f, 1.6f, -191.4f};
float library [] = {255.5f, 1.6f, -222.8f};
float b3 [] = {280.8f, 1.6f, -177.2f};
float canteen [] = {279.1f, 1.6f, -224.1f};
float b6 [] = {347.5f, 1.6f, -204.7f};
float b13 []   {364.1f, 1.6f, -211.1f};
float b14 []   {335.7f, 1.6f, -256.2f};
float b15 []   {414.3f, 1.6f, -263.4f};
float b16 []   {447.6f, 1.6f, -258.8f};
float b17 []   {482.1f, 1.6f, -198.2f};
float b18 []   {472f, 1.6f, -198.2f};
float room31 [] = {260.5f, 1.6f, -165.9f};
float room32 [] = {268.8f, 1.6f, -164.6f};
float admin [] = {282.2f, 1.6f, -146.8f};
float b3north [] = {270.7f, 1.6f, -127.1f};
float room33 [] = {299.6f, 1.6f, -117.3f};
float lib11 [] = {227f, 5.2f, -230f};



float points [][] =  {{286.7f, 1.6f, -191.4f},//guild
                     {255.5f, 1.6f, -222.8f},//lib
                     {280.8f, 1.6f, -177.2f},//b3
                     {279.1f, 1.6f, -224.1f},//can
                     {282.1f, 1.6f, -207.5f},//n1
                     {347.5f, 1.6f, -204.7f},//b6
                     {364.1f, 1.6f, -211.1f},//b13
                     {335.7f, 1.6f, -256.2f},//b14
                     {414.3f, 1.6f, -263.4f},//b15
                     {447.6f, 1.6f, -258.8f},//b16
                     {482.1f, 1.6f, -198.2f},//b17
                     {472f, 1.6f, -198.2f},//b18
                     {420.8f, 1.6f, -207.8f},//n2
                     {420.4f, 1.6f, -246.9f},//n3
                     {336.1f, 1.6f, -207.4f},//n4
                     {336.1f, 1.6f, -246.5f},//n5
                     {247f, 1.6f, -228f},//lib01
                     {244f, 1.6f, -228f},//lib02
                     {240f, 4.1f, -228f},//lib03
                     {237f, 4.1f, -230f},//lib04
                     {235f, 5.2f, -230f},//lib05
                     {227f, 5.2f, -230f}};//lib06



                    //1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2
float edges [][] =  {{0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},//guild*1
                     {0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0},//lib*2
                     {1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},//b3*3
                     {1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},//can*4
                     {1,1,1,1,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},//n1*5
                     {0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0},//b6*6
                     {0,0,0,0,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0},//b13*7
                     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0},//b14*8
                     {0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,0},//b15*9
                     {0,0,0,0,0,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,0,0},//b16*10
                     {0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0},//b17*11
                     {0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0},//b18*12
                     {0,0,0,0,0,0,0,0,0,1,1,1,0,1,1,0,0,0,0,0,0,0},//n2*13
                     {0,0,0,0,0,0,0,0,1,1,0,0,1,0,0,1,0,0,0,0,0,0},//n3*14
                     {0,0,0,0,1,1,1,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0},//n4*15;
                     {0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0},//n5*16
                     {0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0},
                     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0},
                     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0},
                     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0},
                     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1},
                     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0}};

char label[]     = {'a'/*guild*/,'b'/*lib*/,
                     'c'/*b3*/,'d',/*can*/
                     'e'/*n1*/,'f'/*b6*/,'g'/*b13*/,
                     'h'/*b14*/,'i'/*b15*/,'j'/*b16*/,
                     'k'/*b17*/,'l'/*b18*/,'m'/*n2*/,
                     'n'/*n3*/,'o'/*n4*/,'p'/*n5*/,
                     'q','r','s','t','u','v'};

float opoints [][]   {{0f,1f,0f,1.571f},//south
                     {0f,1f,0f,0f},//west
                     {0f,1f,0f,4.712f},//north
                    {0f,1f,0f,3.142f}};//east

//
//any variables used in the data structure are here
//although many of these could have been local
//time ran out to make the code as efficient as could be
//and naming conventions have gone right out the window
//

float fullposvrml[][];
float fullorientvrml[][];
int z = 0;
float opposite;
float zdifference;
float hypotenusea;
float hypotenuseb;
float hypotenuse;
float sinofangle;
float radianangle;
float end[] = {0,0,0};
float [] valpsx = new float [1];
float [] valpsz = new float [1];
float [] valxpos = new float [22];
float [] valzpos = new float [22];
float distax;
float distaz;
float dist;
float [] distArray = new float [22];
float lowestdistance = 9999;
int index = 0;
int indexpath = 0;
char end2;
int a,g,b,c,d,e,f,h,i,j,k,n,endnum,endnum2,indexpath2;
char tempchar;
float valx1pos;
float valz1pos;
float valx3pos;
float valz3pos;
float valx2pos;
float valz2pos;
float dist1;
float dist2;
float finaldist;
float dist1a;
float dist2a;
float finaldista;
float pathdist;
float lowestdist = 9999;
float overalldistance = 0;

//
//vectors used that are a part of the shortest path algorithm
//

Vector masvector = new Vector();
Vector vector0 = new Vector();
Vector vectorfirst0 = new Vector();
Vector vectorend = new Vector();
Vector path = new Vector();

//assorted variables for other functions of the applet

float xvalp = 0f;
float yvalp = 0f;
float zvalp = 0f;
float xvalr = 0f;
float yvalr = 0f;
float zvalr = 0f;
float avalr = 0f;


float daysky [][] = {{0.0f, 0.2f, 0.7f},
                    {0.0f, 0.5f, 1.0f},
                    {1.0f, 1.0f, 1.0f}};

float nightsky [][] = {{0f, 0f, 0f},
                      {0f, 0f, 0f},
                      {0f, 0f, 0f}};


float sun01 [][]  = { {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f},{371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}};

float sun23 [][]                = {{371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f},{371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f}};

float sun45 [][]                = {{371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f},{371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}};

float sun67 [][]                = {{371.2f, 4.652f, 415.1f}, {371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f}, {371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f}};

float sun89 [][]                = { {371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f}, {371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f}};

float sun1011 [][]              = {{371.2f, 594.3f, 23.84f}, {371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f}};

float sun1213 [][]              = { {371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f}};

float sun1415 [][]              = {{371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f}};

float sun1617 [][]              = { {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}};

float sun1819 [][]              = {{371.2f, -175.9f, -860.9f}, {371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f}};

float sun2021 [][]              = {{371.2f, -480.7f, -666.3f},
                                  {371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                                  {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                                  {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                                  {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                                  {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                                  {371.2f, -175.9f, -860.9f}};
```

```
float sun2223 [][]  = {{371.2f, -633.3f, -337.4f}, {371.2f, -585f, 23.84f},
                    {371.2f, -348.5f, 304.4f}, {371.2f, 4.652f, 415.1f},
                    {371.2f, 4.652f, 415.1f},{371.2f, 357.8f, 304.4f},
                    {371.2f, 594.3f, 23.84f},{371.2f, 642.7f, -337.4f},
                    {371.2f, 490f, -666.3f}, {371.2f, 185.3f, -860.9f},
                    {371.2f, -175.9f, -860.9f},{371.2f, -480.7f, -666.3f}};

boolean error = false;

      void button12_Clicked(Event event) {
              (new CREDITS()).show();
      }

      void button8_Clicked(Event event) {
              (new HELP()).show();
      }

      void button4_Clicked(Event event) {
              (new HELP()).show();
      }

      void button1_Clicked(Event event) {
              (new HELP()).show();
      }

      void textField1_EnterHit(Event event) {
              list1.select(textField1.getSelectionStart());
   }

      void textField2_EnterHit(Event event) {
              list2.select(textField1.getSelectionStart());
   }


public void init() {
  //getting a hold the browser and embedded file (VRML)
  //for netscape browsers only

  JSObject win = JSObject.getWindow(this);
  JSObject doc = (JSObject) win.getMember("document");
  JSObject embeds = (JSObject) doc.getMember("embeds");
  browser = (Browser) embeds.getSlot(0);

  try {
    //getting a hold of the VRML nodes
    Node camera = browser.getNode("CAM");
    Node mover = browser.getNode("CAMPOS");
    Node omover = browser.getNode("CAMROT");
    Node sensor1 = browser.getNode("TOUCH_GUILD");
    Node sensor2 = browser.getNode("TOUCH_CANTEEN");
    Node sensor3 = browser.getNode("TOUCH_B14");
    Node sensor4 = browser.getNode("TOUCH_B15");
    Node sensor5 = browser.getNode("TOUCH_B16");
    Node sensor6 = browser.getNode("TOUCH_B17");
    Node sensor7 = browser.getNode("TOUCH_B13");
    Node sensor8 = browser.getNode("TOUCH_B6");
    Node sensor9 = browser.getNode("TOUCH_B18");
    Node ps = browser.getNode("GLOBALPS");
    Node time = browser.getNode("CAMTIME");
    Node otime = browser.getNode("OCAMTIME");
    Node sky = browser.getNode("SKY");
    Node sunpi = browser.getNode("sunpi");
    Node suntime = browser.getNode("suntime");

    //getting a hold of the VRML fields Eventin contained within the nodes

    translation = (EventInSFVec3f) camera.getEventIn("set_position");
    rotate = (EventInSFRotation) camera.getEventIn("set_orientation");
    kv = (EventInMFVec3f) mover.getEventIn("set_keyValue");
    okv = (EventInMFRotation) omover.getEventIn("set_keyValue");
    loopy = (EventInSFBool) time.getEventIn("loop");
    enable = (EventInSFBool) time.getEventIn("enabled");
    stop = (EventInSFTime) time.getEventIn("stopTime");
    start = (EventInSFTime) time.getEventIn("startTime");
    timing = (EventInSFTime) time.getEventIn("cycleInterval");
    oloopy = (EventInSFBool) otime.getEventIn("loop");
    oenable = (EventInSFBool) otime.getEventIn("enabled");
    ostop = (EventInSFTime) otime.getEventIn("stopTime");
    ostart = (EventInSFTime) otime.getEventIn("startTime");
    otiming = (EventInSFTime) otime.getEventIn("cycleInterval");
    skychange = (EventInMFColor) sky.getEventIn("skyColor");
    newsunpi = (EventInMFVec3f) sunpi.getEventIn("set_keyValue");
    newsuntime = (EventInSFBool) suntime.getEventIn("loop");

    //getting a hold of the VRML fields Eventout contained within the nodes
    newtrans = (EventOutSFVec3f) camera.getEventOut("position");
    newrot = (EventOutSFRotation) camera.getEventOut("orientation");
    newkv = (EventOutMFVec3f) mover.getEventOut("keyValue");
    onewkv = (EventOutMFRotation) omover.getEventOut("keyValue");
    touchguild = (EventOutSFBool) sensor1.getEventOut("isOver");
    touchcanteen = (EventOutSFBool) sensor2.getEventOut("isOver");
    touchb14 = (EventOutSFBool) sensor3.getEventOut("isOver");
    touchb15 = (EventOutSFBool) sensor4.getEventOut("isOver");
    touchb16 = (EventOutSFBool) sensor5.getEventOut("isOver");
    touchb17 = (EventOutSFBool) sensor6.getEventOut("isOver");
    touchb13 = (EventOutSFBool) sensor7.getEventOut("isOver");
    touchb6 = (EventOutSFBool) sensor8.getEventOut("isOver");
    touchb18 = (EventOutSFBool) sensor9.getEventOut("isOver");
    globalps = (EventOutSFVec3f) ps.getEventOut("position_changed");
    oglobalps = (EventOutSFRotation) ps.getEventOut("orientation_changed");

    // Set up its callback from the EventOuts
    newtrans.advise(this,new Integer(3));
    newkv.advise(this,new Integer(4));
    onewkv.advise(this,new Integer(16));
    globalps.advise(this,new Integer(5));
    newrot.advise(this,new Integer(15));
    touchguild.advise(this, new Integer(6));
    touchcanteen.advise(this, new Integer(7));
    touchb14.advise(this, new Integer(8));
    touchb15.advise(this, new Integer(9));
    touchb16.advise(this, new Integer(10));
    touchb17.advise(this, new Integer(11));
    touchb13.advise(this, new Integer(12));
    touchb6.advise(this, new Integer(13));
    touchb18.advise(this, new Integer(14));


  }
  catch (InvalidNodeException ne) {
    add(new TextField("Failure in VRML" + ne));
    error = true;
  }
  catch (InvalidEventInException ee) {
    add(new TextField("Failure in VRML" + ee));
    error = true;
  }
  catch (InvalidEventOutException ee) {
    add(new TextField("Failure in VRML" + ee));
    error = true;
  }

            //{{INIT_CONTROLS
            setLayout(null);
            addNotify();
            resize(477,497);
            setForeground(new Color(0));
            setBackground(new Color(16777215));
            tabPanel1 = new symantec.itools.awt.TabPanel();
            tabPanel1.setLayout(null);
            tabPanel1.reshape(0,0,372,396);
            tabPanel1.setForeground(new Color(0));
            tabPanel1.setBackground(new Color(16777215));
            add(tabPanel1);
            {
                   java.lang.String[] tempString = new java.lang.String[8];
                   tempString[0] = new java.lang.String("Extra");
                   tempString[1] = new java.lang.String("Pathfinding");
                   tempString[2] = new java.lang.String("Teleportation");
                   tempString[3] = new java.lang.String("User input");
                   tempString[4] = new java.lang.String("Paths & Map");
                   tempString[5] = new java.lang.String("Imagemap");
                   tempString[6] = new java.lang.String("Touchsensor");
                   tempString[7] = new java.lang.String("Environment");
                   tabPanel1.setPanelLabels(tempString);
            }
            extra = new symantec.itools.awt.BorderPanel();
            GridBagLayout gridBagLayout;
            gridBagLayout = new GridBagLayout();
            extra.setLayout(gridBagLayout);
```

```
extra.reshape(12,33,348,352);
tabPanel1.add(extra);
extra.setLabel("VRML output");
textArea4 = new java.awt.TextArea();
textArea4.reshape(0,124,327,154);
GridBagConstraints gbc;
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 3;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(textArea4, gbc);
extra.add(textArea4);
label14 = new java.awt.Label("Output from VRML, for system testing");
label14.reshape(30,33,266,24);
label14.setFont(new Font("Courier", Font.BOLD, 12));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(label14, gbc);
extra.add(label14);
pf = new symantec.itools.awt.BorderPanel();
gridBagLayout = new GridBagLayout();
pf.setLayout(gridBagLayout);
pf.reshape(12,33,348,352);
tabPanel1.add(pf);
pf.setLabel("pathfinding");
label3 = new java.awt.Label("SEARCH FOR DESTINATION",Label.CENTER);
label3.reshape(90,-2,146,21);
label3.setFont(new Font("Courier", Font.BOLD, 10));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 1;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(label3, gbc);
pf.add(label3);
textField2 = new java.awt.TextField(20);
textField2.reshape(50,18,227,22);
textField2.setFont(new Font("Dialog", Font.BOLD, 8));
textField2.setForeground(new Color(0));
textField2.setBackground(new Color(16777215));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.BOTH;
gbc.insets = new Insets(0,50,10,50);
gridBagLayout.setConstraints(textField2, gbc);
pf.add(textField2);
list2 = new java.awt.List(0,false);
list2.addItem("guild(pf)");
list2.addItem("library(pf)");
list2.addItem("canteen(pf)");
list2.addItem("building 6(pf)");
list2.addItem("building 13(pf)");
list2.addItem("building 3(pf)");
list2.addItem("building 14(pf)");
list2.addItem("building 15(pf)");
list2.addItem("building 16(pf)");
list2.addItem("building 17(pf)");
list2.addItem("building 18(pf)");
list2.addItem("library level1(pf)");
list2.reshape(50,48,227,58);
list2.setForeground(new Color(0));
list2.setBackground(new Color(16777215));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 3;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.BOTH;
gbc.insets = new Insets(0,50,0,50);
gridBagLayout.setConstraints(list2, gbc);
pf.add(list2);
label4 = new java.awt.Label("NEW POSITION",Label.CENTER);
label4.reshape(120,107,86,21);
label4.setFont(new Font("Courier", Font.BOLD, 10));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(label4, gbc);
pf.add(label4);
textArea2 = new java.awt.TextArea();
textArea2.reshape(50,127,227,154);
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 5;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,50,0,50);
gridBagLayout.setConstraints(textArea2, gbc);
pf.add(textArea2);
button8 = new java.awt.Button("help");
button8.reshape(144,290,39,20);
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(10,0,5,0);
gridBagLayout.setConstraints(button8, gbc);
pf.add(button8);
tele = new symantec.itools.awt.BorderPanel();
gridBagLayout = new GridBagLayout();
tele.setLayout(gridBagLayout);
tele.reshape(12,33,348,352);
tabPanel1.add(tele);
tele.setLabel("teleportation");
label1 = new java.awt.Label("SEARCH FOR DESTINATION");
label1.reshape(91,0,0,0);
label1.setFont(new Font("Courier", Font.BOLD, 10));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 1;
gbc.gridwidth = 2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(label1, gbc);
tele.add(label1);
textField1 = new java.awt.TextField(20);
textField1.reshape(50,21,0,0);
textField1.setFont(new Font("Dialog", Font.BOLD, 8));
textField1.setForeground(new Color(0));
textField1.setBackground(new Color(16777215));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.BOTH;
gbc.insets = new Insets(0,50,0,50);
gridBagLayout.setConstraints(textField1, gbc);
tele.add(textField1);
list1 = new java.awt.List(0,false);
list1.addItem("library");
list1.addItem("guild");
list1.addItem("building 3");
list1.addItem("building 6");
list1.addItem("building 13");
list1.addItem("canteen");
list1.addItem("building 14");
list1.addItem("building 15");
list1.addItem("building 16");
```

```
list1.addItem("building 17");
list1.addItem("building 18");
list1.addItem("room 3.1");
list1.addItem("room 3.2");
list1.addItem("administration");
list1.addItem("building 3 north");
list1.addItem("room 3.3");
list1.reshape(50,52,227,32);
list1.setForeground(new Color(0));
list1.setBackground(new Color(16777215));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth   2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.BOTH;
gbc.insets = new Insets(10,50,0,50);
gridBagLayout.setConstraints(list1, gbc);
tele.add(list1);
label2 = new java.awt.Label("NEW POSITION AND ORIENTATION");
label2.reshape(73,112,0,0);
label2.setFont(new Font("Courier", Font.BOLD, 10));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.gridwidth   2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(label2, gbc);
tele.add(label2);
textArea1 = new java.awt.TextArea();
textArea1.setEditable(false);
textArea1.reshape(20,133,287,129);
textArea1.setForeground(new Color(0));
textArea1.setBackground(new Color(16777215));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 5;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,20,5,20);
gridBagLayout.setConstraints(textArea1, gbc);
tele.add(textArea1);
button4 = new java.awt.Button("help");
button4.reshape(144,292,0,0);
button4.setForeground(new Color(0));
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.gridwidth   2;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,0,0,0);
gridBagLayout.setConstraints(button4, gbc);
tele.add(button4);
textArea3 = new java.awt.TextArea();
textArea3.reshape(184,133,124,154);
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 5;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.fill = GridBagConstraints.NONE;
gbc.insets = new Insets(0,20,5,20);
gridBagLayout.setConstraints(textArea3, gbc);
tele.add(textArea3);
uic = new symantec.itools.awt.BorderPanel();
uic.setLayout(null);
uic.reshape(12,33,348,352);
tabPanel1.add(uic);
uic.setLabel("user input coordinate");
label11 = new java.awt.Label("Position",Label.CENTER);
label11.reshape(0,15,85,24);
uic.add(label11);
label12 = new java.awt.Label("Rotation",Label.CENTER);
label12.reshape(228,15,63,21);
```

```
		uic.add(label12);
		label8 = new java.awt.Label("X");
		label8.reshape(36,75,23,21);
		uic.add(label8);
		radioButtonGroupPanel1 = new
		symantec.itools.awt.RadioButtonGroupPanel();
		radioButtonGroupPanel1.setLayout(null);
		radioButtonGroupPanel1.reshape(204,63,109,189);
		uic.add(radioButtonGroupPanel1);
		Group1 = new CheckboxGroup();
		radioButtonnorth = new java.awt.Checkbox("North", Group1, false);
		radioButtonnorth.reshape(12,12,95,21);
		radioButtonGroupPanel1.add(radioButtonnorth);
		radioButtoneast = new java.awt.Checkbox("East", Group1, false);
		radioButtoneast.reshape(12,36,95,21);
		radioButtonGroupPanel1.add(radioButtoneast);
		radioButtonwest = new java.awt.Checkbox("West", Group1, false);
		radioButtonwest.reshape(12,60,95,21);
		radioButtonGroupPanel1.add(radioButtonwest);
		radioButtonsouth = new java.awt.Checkbox("South", Group1, false);
		radioButtonsouth.reshape(12,84,95,21);
		radioButtonGroupPanel1.add(radioButtonsouth);
		xpos = new java.awt.TextField();
		xpos.reshape(84,75,80,21);
		uic.add(xpos);
		label9 = new java.awt.Label("Y");
		label9.reshape(36,123,23,21);
		uic.add(label9);
		ypos = new java.awt.TextField();
		ypos.reshape(84,123,80,21);
		uic.add(ypos);
		label10 = new java.awt.Label("Z");
		label10.reshape(36,171,23,21);
		uic.add(label10);
		zpos = new java.awt.TextField();
		zpos.reshape(84,171,80,21);
		uic.add(zpos);
		button6 = new java.awt.Button("submit");
		button6.reshape(132,279,47,21);
		uic.add(button6);
		verticalLine1 = new symantec.itools.awt.shape.VerticalLine();
		verticalLine1.reshape(204,15,2,230);
		uic.add(verticalLine1);
		imd = new symantec.itools.awt.BorderPanel();
		imd.setLayout(null);
		imd.reshape(12,33,348,352);
		tabPanel1.add(imd);
		imd.setLabel("paths and map");
		reddot1 = new symantec.itools.multimedia.ImageViewer();
		reddot1.reshape(24,39,3,3);
		imd.add(reddot1);
		try {
			reddot1.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/block.GIF"));
		} catch (java.net.MalformedURLException error) {
		}
		imageViewer2 = new symantec.itools.multimedia.ImageViewer();
		imageViewer2.reshape(0,15,320,240);
		imd.add(imageViewer2);
		try {
			imageViewer2.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/birdseye.gif"));
		} catch (java.net.MalformedURLException error) {
		}
		invisibleButtonlibpf = new symantec.itools.awt.InvisibleButton();
		invisibleButtonlibpf.reshape(36,51,39,40);
		imd.add(invisibleButtonlibpf);
		invisibleButtoncanpf = new symantec.itools.awt.InvisibleButton();
		invisibleButtoncanpf.reshape(84,63,31,27);
		imd.add(invisibleButtoncanpf);
		invisibleButtonguildpf = new symantec.itools.awt.InvisibleButton();
		invisibleButtonguildpf.reshape(60,99,17,12);
		imd.add(invisibleButtonguildpf);
		invisibleButtonb13pf = new symantec.itools.awt.InvisibleButton();
		invisibleButtonb13pf.reshape(132,63,48,17);
		imd.add(invisibleButtonb13pf);
		invisibleButtonb3pf = new symantec.itools.awt.InvisibleButton();
		invisibleButtonb3pf.reshape(60,123,78,44);
		imd.add(invisibleButtonb3pf);
		invisibleButtonb6pf = new symantec.itools.awt.InvisibleButton();
		invisibleButtonb6pf.reshape(120,99,26,17);
```

```
                imd.add(invisibleButtonb6pf);
                invisibleButtonbl8pf = new symantec.itools.awt.InvisibleButton();
                invisibleButtonbl8pf.reshape(204,99,42,17);
                imd.add(invisibleButtonbl8pf);
                imc = new symantec.itools.awt.BorderPanel();
                imc.setLayout(null);
                imc.reshape(12,33,348,352);
                tabPanell.add(imc);
                imc.setLabel("imagemap");
                labelimclib = new java.awt.Label("library");
                labelimclib.hide();
                labelimclib.reshape(72,87,36,20);
                labelimclib.setForeground(new Color(16777215));
                labelimclib.setBackground(new Color(0));
                imc.add(labelimclib);
                labelimcb3 = new java.awt.Label("building 3");
                labelimcb3.hide();
                labelimcb3.reshape(144,135,60,17);
                imc.add(labelimcb3);
                labelimcb3.disable();
                labelimcguild = new java.awt.Label("guild");
                labelimcguild.hide();
                labelimcguild.reshape(84,111,30,15);
                imc.add(labelimcguild);
                labelimcguild.disable();
                labelimccan = new java.awt.Label("canteen");
                labelimccan.hide();
                labelimccan.reshape(120,75,48,16);
                imc.add(labelimccan);
                labelimccan.disable();
                labelimcbl4 = new java.awt.Label("building 14");
                labelimcbl4.hide();
                labelimcbl4.reshape(144,51,63,14);
                imc.add(labelimcbl4);
                labelimcbl4.disable();
                labelimcbl5 = new java.awt.Label("building 15");
                labelimcbl5.hide();
                labelimcbl5.reshape(180,39,65,15);
                imc.add(labelimcbl5);
                labelimcbl5.disable();
                labelimcbl6 = new java.awt.Label("building 16");
                labelimcbl6.hide();
                labelimcbl6.reshape(228,27,68,15);
                imc.add(labelimcbl6);
                labelimcbl6.disable();
                labelimcbl3 = new java.awt.Label("building 13");
                labelimcbl3.hide();
                labelimcbl3.reshape(180,75,62,14);
                imc.add(labelimcbl3);
                labelimcbl3.disable();
                labelimcbl7 = new java.awt.Label("building 17");
                labelimcbl7.hide();
                labelimcbl7.reshape(228,63,64,15);
                imc.add(labelimcbl7);
                labelimcbl7.disable();
                labelimcb6 = new java.awt.Label("building 6");
                labelimcb6.hide();
                labelimcb6.reshape(144,111,58,12);
                imc.add(labelimcb6);
                labelimcb6.disable();
                labelimcacad = new java.awt.Label("academy of performing arts");
                labelimcacad.hide();
                labelimcacad.reshape(156,195,155,15);
                imc.add(labelimcacad);
                labelimcacad.disable();
                imageViewer3 = new symantec.itools.multimedia.ImageViewer();
                imageViewer3.reshape(0,15,320,240);
                imc.add(imageViewer3);
                try {
                        imageViewer3.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/birdseye.gif"));
                } catch (java.net.MalformedURLException error) {
                }
                invisibleButtonb3 = new symantec.itools.awt.InvisibleButton();
                invisibleButtonb3.reshape(60,123,75,45);
                imc.add(invisibleButtonb3);
                invisibleButtonlib = new symantec.itools.awt.InvisibleButton();
                invisibleButtonlib.reshape(36,63,37,37);
                imc.add(invisibleButtonlib);
                invisibleButtonguild = new symantec.itools.awt.InvisibleButton();
                invisibleButtonguild.reshape(60,111,17,12);
```

```
		imc.add(invisibleButtonguild);
		invisibleButtoncan = new symantec.itools.awt.InvisibleButton();
		invisibleButtoncan.reshape(84,63,33,29);
		imc.add(invisibleButtoncan);
		invisibleButtonbl4 = new symantec.itools.awt.InvisibleButton();
		invisibleButtonbl4.reshape(108,51,30,16);
		imc.add(invisibleButtonbl4);
		invisibleButtonbl3 = new symantec.itools.awt.InvisibleButton();
		invisibleButtonbl3.reshape(132,75,49,19);
		imc.add(invisibleButtonbl3);
		invisibleButtonbl5 = new symantec.itools.awt.InvisibleButton();
		invisibleButtonbl5.reshape(144,27,27,37);
		imc.add(invisibleButtonbl5);
		invisibleButtonbl6 = new symantec.itools.awt.InvisibleButton();
		invisibleButtonbl6.reshape(180,39,117,18);
		imc.add(invisibleButtonbl6);
		invisibleButtonbl7 = new symantec.itools.awt.InvisibleButton();
		invisibleButtonbl7.reshape(204,75,63,14);
		imc.add(invisibleButtonbl7);
		invisibleButtonb6 = new symantec.itools.awt.InvisibleButton();
		invisibleButtonb6.reshape(120,111,25,17);
		imc.add(invisibleButtonb6);
		invisibleButtonacad = new symantec.itools.awt.InvisibleButton();
		invisibleButtonacad.reshape(60,183,94,45);
		imc.add(invisibleButtonacad);
		label5 = new java.awt.Label("X");
		label5.reshape(36,255,23,21);
		imc.add(label5);
		label6 = new java.awt.Label("Y");
		label6.reshape(144,255,23,21);
		imc.add(label6);
		label7 = new java.awt.Label("Z");
		label7.reshape(252,255,23,21);
		imc.add(label7);
		textField4 = new java.awt.TextField();
		textField4.reshape(108,279,93,24);
		imc.add(textField4);
		textField5 = new java.awt.TextField();
		textField5.reshape(216,279,97,24);
		imc.add(textField5);
		reddot2 = new symantec.itools.multimedia.ImageViewer();
		reddot2.reshape(72,87,3,3);
		imc.add(reddot2);
		try {
			reddot2.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/block.GIF"));
		} catch (java.net.MalformedURLException error) {
		}
		textField3 = new java.awt.TextField();
		textField3.reshape(0,279,96,24);
		imc.add(textField3);
		ts = new symantec.itools.awt.BorderPanel();
		ts.setLayout(null);
		ts.reshape(12,33,348,352);
		tabPanel1.add(ts);
		ts.setLabel("touch sensor");
		imageViewer5 = new symantec.itools.multimedia.ImageViewer();
		imageViewer5.reshape(0,15,320,240);
		ts.add(imageViewer5);
		try {
			imageViewer5.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/birdseye.gif"));
		} catch (java.net.MalformedURLException error) {
		}
		tstext = new java.awt.TextField();
		tstext.reshape(36,279,273,20);
		ts.add(tstext);
		libraryhigh = new symantec.itools.multimedia.ImageViewer();
		libraryhigh.hide();
		libraryhigh.reshape(36,63,42,40);
		ts.add(libraryhigh);
		try {
			libraryhigh.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/library.gif"));
		} catch (java.net.MalformedURLException error) {
		}
		canteenhigh = new symantec.itools.multimedia.ImageViewer();
		canteenhigh.hide();
		canteenhigh.reshape(84,75,34,32);
		ts.add(canteenhigh);
		try {
				canteenhigh.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/canteen.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			guildhigh = new symantec.itools.multimedia.ImageViewer();
			guildhigh.hide();
			guildhigh.reshape(60,111,19,13);
			ts.add(guildhigh);
			try {
				guildhigh.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/guild.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b3high = new symantec.itools.multimedia.ImageViewer();
			b3high.hide();
			b3high.reshape(60,135,78,49);
			ts.add(b3high);
			try {
				b3high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b3.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b13high = new symantec.itools.multimedia.ImageViewer();
			b13high.hide();
			b13high.reshape(120,75,52,22);
			ts.add(b13high);
			try {
				b13high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b13.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b14high = new symantec.itools.multimedia.ImageViewer();
			b14high.hide();
			b14high.reshape(108,51,34,19);
			ts.add(b14high);
			try {
				b14high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b14.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b15high = new symantec.itools.multimedia.ImageViewer();
			b15high.hide();
			b15high.reshape(144,39,30,40);
			ts.add(b15high);
			try {
				b15high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b15.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b16high = new symantec.itools.multimedia.ImageViewer();
			b16high.hide();
			b16high.reshape(180,51,119,22);
			ts.add(b16high);
			try {
				b16high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b16.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b17high = new symantec.itools.multimedia.ImageViewer();
			b17high.hide();
			b17high.reshape(204,87,67,16);
			ts.add(b17high);
			try {
				b17high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b17.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b6high = new symantec.itools.multimedia.ImageViewer();
			b6high.hide();
			b6high.reshape(120,111,28,21);
			ts.add(b6high);
			try {
				b6high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b6.gif"));
			} catch (java.net.MalformedURLException error) {
			}
			b18high = new symantec.itools.multimedia.ImageViewer();
			b18high.hide();
			b18high.reshape(204,111,44,22);
			ts.add(b18high);
			try {
```

```
                        b18high.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/b18.gif"));
                } catch (java.net.MalformedURLException error) {
                }
                reddot3 = new symantec.itools.multimedia.ImageViewer();
                reddot3.reshape(36,75,3,3);
                ts.add(reddot3);
                try {
                        reddot3.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/block.GIF"));
                } catch (java.net.MalformedURLException error) {
                }
                env = new symantec.itools.awt.BorderPanel();
                env.setLayout(null);
                env.reshape(12,33,348,352);
                tabPanel1.add(env);
                env.setLabel("environment");
                button5 = new java.awt.Button("time");
                button5.reshape(36,207,38,21);
                env.add(button5);
                button12 = new java.awt.Button("credits");
                button12.reshape(228,255,60,21);
                env.add(button12);
                textField6 = new java.awt.TextField();
                textField6.setEditable(false);
                textField6.reshape(0,255,132,21);
                env.add(textField6);
                label15 = new java.awt.Label("X :",Label.CENTER);
                label15.reshape(24,39,20,20);
                env.add(label15);
                label16 = new java.awt.Label("Y :",Label.CENTER);
                label16.reshape(24,75,20,20);
                env.add(label16);
                label17 = new java.awt.Label("Z :",Label.CENTER);
                label17.reshape(24,111,20,20);
                env.add(label17);
                button1 = new java.awt.Button("help");
                button1.reshape(228,207,59,21);
                env.add(button1);
                label18 = new java.awt.Label("xpos");
                label18.reshape(48,39,80,20);
                env.add(label18);
                label19 = new java.awt.Label("ypos");
                label19.reshape(48,75,80,20);
                env.add(label19);
                label20 = new java.awt.Label("zpos");
                label20.reshape(48,111,80,20);
                env.add(label20);
                label21 = new java.awt.Label("building:");
                label21.reshape(24,159,50,20);
                env.add(label21);
                label22 = new java.awt.Label("outside");
                label22.reshape(84,159,100,20);
                env.add(label22);
                eastp = new symantec.itools.multimedia.ImageViewer();
                eastp.reshape(180,27,122,107);
                env.add(eastp);
                try {
                        eastp.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/east.gif"));
                } catch (java.net.MalformedURLException error) {
                }
                northp = new symantec.itools.multimedia.ImageViewer();
                northp.hide();
                northp.reshape(180,27,122,107);
                env.add(northp);
                try {
                        northp.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/north.gif"));
                } catch (java.net.MalformedURLException error) {
                }
                southp = new symantec.itools.multimedia.ImageViewer();
                southp.reshape(180,27,122,107);
                env.add(southp);
                try {
                        southp.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/south.gif"));
                } catch (java.net.MalformedURLException error) {
                }
                westp = new symantec.itools.multimedia.ImageViewer();
                westp.hide();
```

```
               westp.reshape(180,27,122,107);
               env.add(westp);
               try {
                      westp.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/west.gif"));
               } catch (java.net.MalformedURLException error) {
               }
               //}}
               for (int i = 1; i<=2; i++) {
               Date d = new Date ();
            int h = d.getHours();

            float[][] vals = new float[25][3];
            float [][] valsk = new float [3][3];
               if (h == 0 || h == 1)
                {vals = sun01;
                valsk = nightsky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 2 || h == 3)
                {vals = sun23;
                valsk = nightsky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 4 || h == 5)
                {vals = sun45;
                valsk = nightsky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 6 || h == 7)
                {vals = sun67;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 8 || h == 9)
                {vals = sun89;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 10 || h == 11)
                {vals = sun1011;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 12 || h == 13)
                {vals = sun1213;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 14 || h == 15)
                {vals = sun1415;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 16 || h == 17)
                {vals = sun1617;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 18 || h == 19)
                {vals = sun1819;
                valsk = daysky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 20 || h == 21)
                {vals = sun2021;
                valsk = nightsky;
            newsunpi.setValue(vals);
            skychange.setValue(valsk);}

                else if (h == 22 || h == 23)
```

```
        {vals = sun2223;
        valsk = nightsky;
        newsunpi.setValue(vals);
        skychange.setValue(valsk);
      }
  }


public void paint (Graphics g)
  {
    imageViewer3.repaint();
    reddot2.repaint();
   }


 public void callback(EventOut who, double when, Object which) {
  Integer whichNum = (Integer) which;

  if (whichNum.intValue() == 6) {
   tstext.setText("guild");
   guildhigh.show();
   canteenhigh.hide();
   b14high.hide();
   b15high.hide();
   b16high.hide();
   b17high.hide();
   b13high.hide();
   b6high.hide();
   b18high.hide();
   }
  if (whichNum.intValue() == 7)  {
   tstext.setText("canteen");
   canteenhigh.show();
   guildhigh.hide();
   b14high.hide();
   b15high.hide();
   b16high.hide();
   b17high.hide();
   b13high.hide();
   b6high.hide();
   b18high.hide();
   }
  if (whichNum.intValue() == 8) {
   tstext.setText("b14");
   b14high.show();
   guildhigh.hide();
   canteenhigh.hide();
   b15high.hide();
   b16high.hide();
   b17high.hide();
   b13high.hide();
   b6high.hide();
   b18high.hide();
   }
  if (whichNum.intValue() == 9) {
   tstext.setText("b15");
   b15high.show();
   guildhigh.hide();
   canteenhigh.hide();
   b14high.hide();
   b16high.hide();
   b17high.hide();
   b13high.hide();
   b6high.hide();
   b18high.hide();
   }
  if (whichNum.intValue() == 10) {
   tstext.setText("b16");
   b16high.show();
   guildhigh.hide();
   canteenhigh.hide();
   b14high.hide();
   b15high.hide();
   b17high.hide();
   b13high.hide();
   b6high.hide();
   b18high.hide();
   }
  if (whichNum.intValue() == 11) {
   tstext.setText("b17");
   b17high.show();
```

```
        guildhigh.hide();
        canteenhigh.hide();
        b14high.hide();
        b15high.hide();
        b16high.hide();
        b13high.hide();
        b6high.hide();
        b18high.hide();
        }
      if (whichNum.intValue() == 12) {
        tstext.setText("b13");
        b13high.show();
        guildhigh.hide();
        canteenhigh.hide();
        b14high.hide();
        b15high.hide();
        b16high.hide();
        b17high.hide();
        b6high.hide();
        b18high.hide();
        }
      if (whichNum.intValue() == 13) {
        tstext.setText("b6");
        b6high.show();
        guildhigh.hide();
        canteenhigh.hide();
        b14high.hide();
        b15high.hide();
        b16high.hide();
        b17high.hide();
        b13high.hide();
        b18high.hide();
        }
      if (whichNum.intValue() == 14) {
        tstext.setText("b18");
        b18high.show();
        guildhigh.hide();
        canteenhigh.hide();
        b14high.hide();
        b15high.hide();
        b16high.hide();
        b17high.hide();
        b13high.hide();
        b6high.hide();
        }



      if (whichNum.intValue() == 3) {
        float[] val = newtrans.getValue();
        if ((int)end[0] == (int)val[0]
             && (int)end[2] == (int)val[2]
             )
               {
               enable.setValue(false);
               loopy.setValue(false);
               oenable.setValue(false);
               oloopy.setValue(false);
               textArea2.appendText("arrived" + "\n");
               }
               else if ((int)fullposvrml[z+1][0] == (int)val[0]
                      && fullposvrml[z+1][1] == val[1]
                      && (int)fullposvrml[z+1][2] == (int)val[2])
                      {
                          enable.setValue(false);
                          loopy.setValue(false);
                          orientationinterpolator();
                      }
             else {}

        float[] val2 = newrot.getValue();
        textArea1.appendText("x " + (val[0]) + ", " + "y " + val[1] + ", " + "z "
            + Math.floor(val[2]) + "\n" );
            textArea3.appendText(val2[0] + ", " + val2[1] + ", " + val2[2] + ", " +
val2[3] + "\n");
        }

      if (whichNum.intValue() == 5) {
        float [] valps = globalps.getValue();
        textField3.setText("\n" + (int)valps[0]);
            textField4.setText("\n" + valps[1]);
        textField5.setText("\n" + (int)valps[2]);
        label18.setText("" + (int)valps[0]);
        label19.setText("" + valps[1]);
        label20.setText("" + (int)valps[2]);
        reddot1.reshape((int)valps[0]-195,(int)valps[2]+310,3,3);
        reddot1.repaint();
        reddot2.reshape((int)valps[0]-195,(int)valps[2]+310,3,3);
        reddot2.repaint();
        reddot3.reshape((int)valps[0]-195,(int)valps[2]+310,3,3);
        reddot3.repaint();
       }


    if (whichNum.intValue() == 15) {
      float[] valr = newrot.getValue();
      if ( valr[3] < fullorientvrml[z+1][3]+.3
          && valr[3] > fullorientvrml[z+1][3]-.3)
       {
           z++;
           oenable.setValue(false);
           oloopy.setValue(false);
           positioninterpolator();
       }

       else if ((int) valr [3] == -4)
        {
           eastp.hide();
           northp.hide();
           westp.hide();
           southp.show();
           southp.repaint();
        }
        else if ((int)valr [3] == -3)
        {
           southp.hide();
           northp.hide();
           westp.hide();
           eastp.show();
           eastp.repaint();
        }
        else if ((int)valr [3] == -1)
        {
           southp.hide();
           eastp.hide();
           westp.hide();
           northp.show();
           northp.repaint();
        }
        else if (valr [3] == 0)
        {
           southp.hide();
           eastp.hide();
           northp.hide();
           westp.show();
           westp.repaint();
        }
        else {}

    }
    if (whichNum.intValue() == 16)
    {
        float[][] valr = onewkv.getValue();
        textArea4.appendText("onewkv" +"\n" +
          valr[0][0] + " " +valr[0][1] + " " + valr[0][2] + " " + valr[0][3] + "\n" +
          valr[1][0] + " " +valr[1][1] + " " + valr[1][2] + " " + valr[1][3] + "\n" );
    }
    if (whichNum.intValue() == 4)
    {
        float[][] valr = newkv.getValue();
        textArea4.appendText("posnewkv" +"\n" +
                       valr[0][0] + " " +valr[0][1] + " " + valr[0][2] +  "\n" +
                       valr[1][0] + " " +valr[1][1] + " " + valr[1][2] +  "\n" );

    }
}

public boolean handleEvent(Event event) {
               if (event.target == button4 && event.id == Event.ACTION_EVENT) {
                       button4_Clicked(event);
                       return true;
               }
        if (event.target == button8 && event.id == Event.ACTION_EVENT) {
                button8_Clicked(event);
                return true;
        }

        if (event.target == button12 && event.id == Event.ACTION_EVENT) {
                button12_Clicked(event);
                return true;
        }

        if (event.target == button1 && event.id == Event.ACTION_EVENT) {
                button1_Clicked(event);
                return true;
        }

if (event.target == invisibleButtonlib && event.id == Event.MOUSE_ENTER) {
                invisibleButtonlib_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonlib && event.id == Event.MOUSE_EXIT) {
                invisibleButtonlib_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonb3 && event.id == Event.MOUSE_ENTER) {
                invisibleButtonb3_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonb3 && event.id == Event.MOUSE_EXIT) {
                invisibleButtonb3_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonguild && event.id == Event.MOUSE_ENTER) {
                invisibleButtonguild_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonguild && event.id == Event.MOUSE_EXIT) {
                invisibleButtonguild_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtoncan && event.id == Event.MOUSE_ENTER) {
                invisibleButtoncan_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtoncan && event.id == Event.MOUSE_EXIT) {
                invisibleButtoncan_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonb14 && event.id == Event.MOUSE_ENTER) {
                invisibleButtonb14_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonb14 && event.id == Event.MOUSE_EXIT) {
                invisibleButtonb14_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonb13pf && event.id == Event.MOUSE_ENTER) {
                invisibleButtonb13_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonb13pf && event.id == Event.MOUSE_EXIT) {
                invisibleButtonb13_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonb15 && event.id == Event.MOUSE_ENTER) {
                invisibleButtonb15_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonb15 && event.id == Event.MOUSE_EXIT) {
                invisibleButtonb15_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonb16 && event.id == Event.MOUSE_ENTER) {
                invisibleButtonb16_MouseEnter(event);
                return true;
        }
if (event.target == invisibleButtonb16 && event.id == Event.MOUSE_EXIT) {
                invisibleButtonb16_MouseExit(event);
                return true;
        }
if (event.target == invisibleButtonb17 && event.id == Event.MOUSE_ENTER) {
```

```
                    invisibleButtonb17_MouseEnter(event);
                    return true;
            }
    if (event.target == invisibleButtonb17 && event.id == Event.MOUSE_EXIT) {
                    invisibleButtonb17_MouseExit(event);
                    return true;
            }
    if (event.target == invisibleButtonb13pf && event.id== Event.MOUSE_ENTER){
                    invisibleButtonb6_MouseEnter(event);
                    return true;
            }
    if (event.target == invisibleButtonb13pf && event.id== Event.MOUSE_EXIT) {
                    invisibleButtonb6_MouseExit(event);
                    return true;
            }
    if (event.target == invisibleButtonacad && event.id == Event.MOUSE_ENTER) {
                    invisibleButtonacad_MouseEnter(event);
                    return true;
            }
    if (event.target == invisibleButtonacad && event.id == Event.MOUSE_EXIT) {
                    invisibleButtonacad_MouseExit(event);
                    return true;
            }

    if (event.target == invisibleButtonb3 && event.id== Event.ACTION_EVENT) {
                    invisibleButtonb3_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonlib && event.id == Event.ACTION_EVENT) {
                    invisibleButtonlib_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonguild && event.id == Event.ACTION_EVENT) {
                    invisibleButtonguild_Action(event);
                    return true;
            }
    if (event.target == invisibleButtoncan && event.id == Event.ACTION_EVENT) {
                    invisibleButtoncan_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonb14 && event.id == Event.ACTION_EVENT) {
                    invisibleButtonb14_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonb13pf && event.id == Event.ACTION_EVENT) {
                    invisibleButtonb13_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonb15 && event.id == Event.ACTION_EVENT) {
                    invisibleButtonb15_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonb16 && event.id == Event.ACTION_EVENT) {
                    invisibleButtonb16_Action(event);
                    return true;
            }
    if (event.target == invisibleButtonb17 && event.id == Event.ACTION_EVENT) {
                    invisibleButtonb17_Action(event);
                    return true;
            }
            if (event.target == textField1 && event.id ==Event.ACTION_EVENT) {
                    textField1_EnterHit(event);
                    return true;
            }
            if (event.target == textField2 && event.id ==Event.ACTION_EVENT) {
                    textField2_EnterHit(event);
                    return true;
            }
            return super.handleEvent(event);
    }

void invisibleButtonb3_MouseExit(Event event) {
            labelimcb3.hide();
            imageViewer3.repaint();
    }
    void invisibleButtonb3_MouseEnter(Event event) {
            labelimcb3.show();
    }
void invisibleButtonlib_MouseExit(Event event) {
            labelimclib.hide();
            imageViewer3.repaint();
```

```
}
void invisibleButtonlib_MouseEnter(Event event) {
        labelimclib.show();
}
void invisibleButtonguild_MouseExit(Event event) {
        labelimcguild.hide();
        imageViewer3.repaint();
}
void invisibleButtonguild_MouseEnter(Event event) {
        labelimcguild.show();
}
void invisibleButtoncan_MouseExit(Event event) {
        labelimccan.hide();
        imageViewer3.repaint();
}
void invisibleButtoncan_MouseEnter(Event event) {
        labelimccan.show();
}
void invisibleButtonb14_MouseExit(Event event) {
        labelimcb14.hide();
        imageViewer3.repaint();
}
void invisibleButtonb14_MouseEnter(Event event) {
        labelimcb14.show();
}
void invisibleButtonb13_MouseExit(Event event) {
        labelimcb13.hide();
        imageViewer3.repaint();
}
void invisibleButtonb13_MouseEnter(Event event) {
        labelimcb13.show();
}
void invisibleButtonb15_MouseExit(Event event) {
        labelimcb15.hide();
        imageViewer3.repaint();
}
void invisibleButtonb15_MouseEnter(Event event) {
        labelimcb15.show();
}
void invisibleButtonb16_MouseExit(Event event) {
        labelimcb16.hide();
        imageViewer3.repaint();
}
void invisibleButtonb16_MouseEnter(Event event) {
        labelimcb16.show();
}
void invisibleButtonb17_MouseExit(Event event) {
        labelimcb17.hide();
        imageViewer3.repaint();
}
void invisibleButtonb17_MouseEnter(Event event) {
        labelimcb17.show();
}
void invisibleButtonb6_MouseExit(Event event) {
        labelimcb6.hide();
        imageViewer3.repaint();
}
void invisibleButtonb6_MouseEnter(Event event) {
        labelimcb6.show();
}
void invisibleButtonacad_MouseExit(Event event) {
        labelimcacad.hide();
        imageViewer3.repaint();
}
void invisibleButtonacad_MouseEnter(Event event) {
        labelimcacad.show();
}
void invisibleButtonb3_Action(Event event) {
 teleportation(b3,east);
}
void invisibleButtonb17_Action(Event event) {
        teleportation(b17,west);
}
void invisibleButtonb16_Action(Event event) {
        teleportation(b16,west);
}
void invisibleButtonb15_Action(Event event) {
        teleportation(b15,south);
}
void invisibleButtonb13_Action(Event event) {
        teleportation(b13,west);
```

```
      }
      void invisibleButtonb14_Action(Event event) {
              teleportation(b14,west);
      }
      void invisibleButtoncan_Action(Event event) {
              teleportation(canteen,north);
      }
      void invisibleButtonguild_Action(Event event) {
              teleportation(guild,south);
      }
      void invisibleButtonlib_Action(Event event) {
              teleportation(library,south);
      }
      void invisibleButtonlibpf_Action(Event event) {
       end[0] = library[0];
       end[1] = library[1];
       end[2] = library[2];
       nearestneighbour (label[1],1);
      }
      void invisibleButtoncanpf_Action(Event event) {
              end[0] = canteen[0];
       end[1] = canteen[1];
       end[2] = canteen[2];
       nearestneighbour (label[3],3);
      }
      void invisibleButtonguildpf_Action(Event event) {
              end[0] = guild[0];
       end[1] = guild[1];
       end[2] = guild[2];
       nearestneighbour (label[0],0);
      }
      void invisibleButtonb13pf_Action(Event event) {
              end[0] = b13[0];
       end[1] = b13[1];
       end[2] = b13[2];
       nearestneighbour (label[6],6);
      }
      void invisibleButtonb3pf_Action(Event event) {
              end[0] = b3[0];
       end[1] = b3[1];
       end[2] = b3[2];
       nearestneighbour (label[2],2);
      }
      void invisibleButtonb6pf_Action(Event event) {
              end[0] = b6[0];
       end[1] = b6[1];
       end[2] = b6[2];
       nearestneighbour (label[5],5);
      }
      void invisibleButtonb18pf_Action(Event event) {
              end[0] = b18[0];
       end[1] = b18[1];
       end[2] = b18[2];
       nearestneighbour (label[11],11);
      }


   public void teleportation(float[] telepos, float[] telerot)
     {
         translation.setValue(telepos);
         rotate.setValue(telerot);
     }



public boolean action(Event event, Object what) {

   if (event.target instanceof Button)
     {
       Button b = (Button) event.target;

       if (b.getLabel() == "time")
       {
           Date d = new Date ();
           textField6.setText (d.toString ());
   }

   else if (b.getLabel() == "submit")
   {
        xvalp  = Float.valueOf(xpos.getText()).floatValue();
        yvalp  = Float.valueOf(ypos.getText()).floatValue();
        zvalp  = Float.valueOf(zpos.getText()).floatValue();
        float[] valp = new float[3];
        valp[0] = xvalp;
        valp[1] = yvalp;
        valp[2] = zvalp;
        translation.setValue(valp);

           if (event.target instanceof Checkbox)
           {
               if (radioButtonnorth.getState() == true)
               {
                rotate.setValue(north);
               }
               else if (radioButtonwest.getState() == true)
               {
                rotate.setValue(west);
               }
               else if (radioButtoneast.getState() == true)
               {
                rotate.setValue(east);
               }
               else if (radioButtonsouth.getState() == true)
               {
                rotate.setValue(south);
               }
           }
   }
   return true;
}

         if (event.target instanceof List)

     {
       List l = (List) event.target;

        if (l.getSelectedItem() == "guild")
       {
          teleportation(guild,south);
       }

        else if (l.getSelectedItem() == "building 3")
       {
          teleportation(b3,east);
       }

        else if (l.getSelectedItem() == "library")
       {
          teleportation(library,south);
       }

           else if (l.getSelectedItem() == "canteen")
       {
          teleportation(canteen,north);
       }

        else if (l.getSelectedItem() == "building 6")
       {
          teleportation(b6,east);
       }

        else if (l.getSelectedItem() == "building 13")
       {
          teleportation(b13,west);
       }

        else if (l.getSelectedItem() == "building 14")
       {
          teleportation(b14,west);
       }

        else if (l.getSelectedItem() == "building 15")
       {
          teleportation(b15,south);
       }

        else if (l.getSelectedItem() == "building 16")
       {
          teleportation(b16,west);
       }
```

```
 else if (l.getSelectedItem() == "building 17")
{
    teleportation(b17,west);
}

 else if (l.getSelectedItem() == "building 18")
{
    teleportation(b18,east);
}

 else if (l.getSelectedItem() == "room 3.1")
{
    teleportation(room31,west);
}

 else if (l.getSelectedItem() == "room 3.2")
{
    teleportation(room32,east);
}

 else if (l.getSelectedItem() == "administration")
{
    teleportation(admin,north);
}

 else if (l.getSelectedItem() == "building 3 north")
{
    teleportation(b3north,north);
}

 else if (l.getSelectedItem() == "room 3.3")
{
    teleportation(room33,west);
}

 else if (l.getSelectedItem() == "guild(pf)")
{
    end[0] = guild[0];
    end[1] = guild[1];
    end[2] = guild[2];
    nearestneighbour (label[0],0);
}

else if (l.getSelectedItem() == "library(pf)")
{
    end[0] = library[0];
    end[1] = library[1];
    end[2] = library[2];
    nearestneighbour (label[1],1);
}

else if (l.getSelectedItem() == "building 3(pf)")
{
    end[0] = b3[0];
    end[1] = b3[1];
    end[2] = b3[2];
    nearestneighbour (label[2],2);
}

else if (l.getSelectedItem() == "canteen(pf)")
{
    end[0] = canteen[0];
    end[1] = canteen[1];
    end[2] = canteen[2];
    nearestneighbour (label[3],3);
}

else if (l.getSelectedItem() == "building 6(pf)")
{
    end[0] = b6[0];
    end[1] = b6[1];
    end[2] = b6[2];
    nearestneighbour (label[5],5);
}

else if (l.getSelectedItem() == "building 13(pf)")
{
    end[0] = b13[0];
    end[1] = b13[1];
    end[2] = b13[2];
    nearestneighbour (label[6],6);
```

```
        }

        else if (l.getSelectedItem() == "building 14(pf)")
        {
            end[0] = b14[0];
            end[1] = b14[1];
            end[2] = b14[2];
            nearestneighbour (label[7],7);
        }

        else if (l.getSelectedItem() == "building 15(pf)")
        {
            end[0] = b15[0];
            end[1] = b15[1];
            end[2] = b15[2];
            nearestneighbour (label[8],8);
        }

        else if (l.getSelectedItem() == "building 16(pf)")
        {
            end[0] = b16[0];
            end[1] = b16[1];
            end[2] = b16[2];
            nearestneighbour (label[9],9);
        }

        else if (l.getSelectedItem() == "building 17(pf)")
        {
            end[0] = b17[0];
            end[1] = b17[1];
            end[2] = b17[2];
            nearestneighbour (label[10],10);
        }

        else if (l.getSelectedItem() == "building 18(pf)")
        {
            end[0] = b18[0];
            end[1] = b18[1];
            end[2] = b18[2];
            nearestneighbour (label[11],11);
        }

        else if (l.getSelectedItem() == "library level1(pf)")
        {
            end[0] = libl1[0];
            end[1] = libl1[1];
            end[2] = libl1[2];
            nearestneighbour (label[21],21);
        }

      }
    return true;
  }

  //
  //
  //the pathfinding function begins here
  //their is room for a lot of improvement
  //in the efficiency of this code, but as it
  //stands it works fine, just not very quickly.
  //
  //

  public void nearestneighbour (char end,int endnum)
  {     textArea2.appendText("finding shortest path," + "\n" +
                             "please wait...." + "\n");
        end2 = end;
        endnum2 = endnum;
        float [] vals = globalps.getValue();
        valpsx[0] = vals[0];
        valpsz[0] = vals[2];

        for (i = 0;i <= 21; i++)
                {
                valxpos[i] = points[i][0];
                valzpos[i] = points[i][2];
                distax = (valxpos[i] - valpsx[0]) * (valxpos[i] - valpsx[0]);
                distaz = (valzpos[i] - valpsz[0]) * (valzpos[i] - valpsz[0]);
                dist = (float)Math.sqrt(distax + distaz);
                distArray [i] = dist;
                    if (distArray[i] < lowestdistance)
                {
                 lowestdistance = distArray[i];
                 index = i;
                 indexpath = i;
                }
            }
            masvector.removeAllElements();
            vector0.removeAllElements();
            vectorfirst0.removeAllElements();
            vectorend.removeAllElements();
            path.removeAllElements();
            vectorfirst0.addElement(new Character(label[indexpath]));
            lowestdistance = 9999;
            lowestdist = 9999;
            overalldistance = 0;
            z = 0;
            if (label[indexpath] != end2)
                {
                    findpath();
                }
                else {}
}

  public void findpath(){

   valx1pos = points [indexpath][0];
   valz1pos = points [indexpath][2];
   valx3pos = points [endnum2][0];
   valz3pos = points [endnum2][2];

   for (g = 0; g<=21; g++)
            {
                if(edges[indexpath][g] >= 1)
                    {
                     vector0 = (Vector)vectorfirst0.clone();

                     if (vector0.lastElement().toString().charAt(0) != label[g]
                         && !vector0.contains(new Character(label[g])))
                     {
                       vector0.addElement(new Character(label[g]));
                       valx2pos = points[g][0];
                       valz2pos = points[g][2];
                       dist1 = (valx2pos - valx1pos) * (valx2pos - valx1pos);
                       dist2 = (valz2pos - valz1pos) * (valz2pos - valz1pos);
                       finaldist = (float)Math.sqrt(dist1 + dist2);
                       dist1a = (valx3pos - valx2pos) * (valx3pos - valx2pos);
                       dist2a = (valz3pos - valz2pos) * (valz3pos - valz2pos);
                       finaldista = (float)Math.sqrt(dist1a + dist2a);
                       pathdist = finaldista + finaldist;

                     if (pathdist < lowestdist)
                        {
                         lowestdist = pathdist;
                         overalldistance +=finaldist;
                         masvector.insertElementAt(vector0,0);
                         indexpath2 = g;
                        }
                         else
                         {
                            masvector.addElement(vector0);
                         }
                      }
                    }
                    else
                    {}
            }
            vectorend = (Vector)masvector.firstElement();

            if (vectorend.lastElement().toString().charAt(0) == end2)
                     {
                      javatovrml();
                     }
                     else
                     {
                      indexpath = indexpath2;
                      vectorfirst0=(Vector)(masvector.firstElement());
                      masvector.removeElementAt(0);
                      lowestdist = 9999;
                      findpath();
                     }
}
```

```
    public void javatovrml(){
                            path = (Vector)masvector.firstElement();
                            textArea2.appendText("shortest path  " + path + "\n");
                            textArea2.appendText("path distance  " + overalldistance +
" mtr" + "\n");
                            textArea2.appendText("estimated time to complete  " +
((overalldistance/c) + ((c-1)*5))
                                                 + " secs" + "\n");
                            c = path.size()-1;
                            float positionarray[][]= new float[path.size()+1][3];
                            float rotationarray[][]= new float[path.size()+1][4];
                            float [] vals = globalps.getValue();
                            float [] ovals = oglobalps.getValue();
                            positionarray[0][0] = vals[0];
                            positionarray[0][1] = vals[1];
                            positionarray[0][2] = vals[2];
                            rotationarray[0][0] = ovals[0];
                            rotationarray[0][1] = ovals[1];
                            rotationarray[0][2] = ovals[2];
                            rotationarray[0][3] = ovals[3];
                            e = 1;

                                for (b = 0; b<=c; b++)
                                    {
                                    tempchar = path.elementAt(b).toString().charAt(0);

                                     for (d = 0; d<=21; d++)
                                        {
                                            if (label[d] == tempchar)
                                            {
                                                 positionarray [e][0] = points[d][0];
                                                 positionarray [e][1] = points[d][1];
                                                 positionarray [e][2] = points[d][2];
                                                 rotationarray [e][0] = 0;
                                                 rotationarray [e][1] = 1;
                                                 rotationarray [e][2] = 0;
                                              }
                                            else {}
                                        }
                                        e++;
                                    }
                                    for (n = 1; n<=c; n++)
                                    {
            opposite = (positionarray [n][0]) - (positionarray [n+1][0]);
            zdifference = (positionarray [n][2]) - (positionarray [n+1][2]);
            hypotenusea = (opposite) * (opposite);
            hypotenuseb = (zdifference) * (zdifference);
            hypotenuse = (float)Math.sqrt(hypotenusea + hypotenuseb);
            sinofangle = (float)Math.sqrt(hypotenusea)/hypotenuse;
            radianangle = (float)Math.asin(sinofangle);
                        if (opposite <= 0 && zdifference <= 0)
                        {
                              rotationarray[n][3] = radianangle + opoints[3][3];
                        }
                              else if (opposite <= 0 && zdifference >= 0)
                         {
                              rotationarray[n][3] = radianangle + opoints[2][3];;
                         }
                              else if (opposite >= 0 && zdifference <= 0)
                         {
                              rotationarray[n][3] = radianangle + opoints[0][3];
                         }
                              else if (opposite >= 0 && zdifference >= 0)
                         {
                              rotationarray[n][3] = radianangle;
                         }
                       }
                textArea2.appendText("sending path to VRML" + "\n");
                fullposvrml = positionarray;
                fullorientvrml = rotationarray;
                positioninterpolator();
        }

    public void orientationinterpolator()
        {
            if (z <= c)
            {
            float[][] orientvrml = new float [2][4];
            orientvrml[0][0] = fullorientvrml[z][0];
            orientvrml[0][1] = fullorientvrml[z][1];
        orientvrml[0][2] = fullorientvrml[z][2];
        orientvrml[0][3] = fullorientvrml[z][3];
        orientvrml[1][0] = fullorientvrml[z+1][0];
        orientvrml[1][1] = fullorientvrml[z+1][1];
        orientvrml[1][2] = fullorientvrml[z+1][2];
        orientvrml[1][3] = fullorientvrml[z+1][3];
        okv.setValue(orientvrml);
        oenable.setValue(true);
        long secs = System.currentTimeMillis()/1000;
        long ostarttime = secs + 1;
        long ostoptime = ostarttime + ((long)10);
        oloopy.setValue(true);
        ostart.setValue((double)ostarttime);
        ostop.setValue((double)ostoptime);
        }
    }

public void positioninterpolator()
    {
        if (z <= c)
        {
            float[][] positionvrml = new float [2][3];
            positionvrml[0][0] = fullposvrml[z][0];
            positionvrml[0][1] = fullposvrml[z][1];
            positionvrml[0][2] = fullposvrml[z][2];
            positionvrml[1][0] = fullposvrml[z+1][0];
            positionvrml[1][1] = fullposvrml[z+1][1];
            positionvrml[1][2] = fullposvrml[z+1][2];
            kv.setValue(positionvrml);
            timing.setValue(overalldistance/c);
            enable.setValue(true);
            long secs = System.currentTimeMillis()/1000;
            long starttime = secs + 1;
            long stoptime = starttime + ((long)overalldistance/c);
            loopy.setValue(true);
            start.setValue((double)starttime);
            stop.setValue((double)stoptime);
        }
    }


    //{{DECLARE_CONTROLS
    symantec.itools.awt.TabPanel tabPanel1;
    symantec.itools.awt.BorderPanel extra;
    java.awt.TextArea textArea4;
    java.awt.Label label14;
    symantec.itools.awt.BorderPanel pf;
    java.awt.Label label3;
    java.awt.TextField textField2;
    java.awt.List list2;
    java.awt.Label label4;
    java.awt.TextArea textArea2;
    java.awt.Button button8;
    symantec.itools.awt.BorderPanel tele;
    java.awt.Label label1;
    java.awt.TextField textField1;
    java.awt.List list1;
    java.awt.Label label2;
    java.awt.TextArea textArea1;
    java.awt.Button button4;
    java.awt.TextArea textArea3;
    symantec.itools.awt.BorderPanel uic;
    java.awt.Label label11;
    java.awt.Label label12;
    java.awt.Label label8;
    symantec.itools.awt.RadioButtonGroupPanel radioButtonGroupPanel1;
    java.awt.Checkbox radioButtonnorth;
    CheckboxGroup Group1;
    java.awt.Checkbox radioButtoneast;
    java.awt.Checkbox radioButtonwest;
    java.awt.Checkbox radioButtonsouth;
    java.awt.TextField xpos;
    java.awt.Label label9;
    java.awt.TextField ypos;
    java.awt.Label label10;
    java.awt.TextField zpos;
    java.awt.Button button6;
    symantec.itools.awt.shape.VerticalLine verticalLine1;
    symantec.itools.awt.BorderPanel imd;
    symantec.itools.multimedia.ImageViewer reddot1;
    symantec.itools.multimedia.ImageViewer imageViewer2;
```

```
        symantec.itools.awt.InvisibleButton invisibleButtonlibpf;
        symantec.itools.awt.InvisibleButton invisibleButtoncanpf;
        symantec.itools.awt.InvisibleButton invisibleButtonguildpf;
        symantec.itools.awt.InvisibleButton invisibleButtonbl3pf;
        symantec.itools.awt.InvisibleButton invisibleButtonb3pf;
        symantec.itools.awt.InvisibleButton invisibleButtonb6pf;
        symantec.itools.awt.InvisibleButton invisibleButtonbl8pf;
        symantec.itools.awt.BorderPanel imc;
        java.awt.Label labelimclib;
        java.awt.Label labelimcb3;
        java.awt.Label labelimcguild;
        java.awt.Label labelimccan;
        java.awt.Label labelimcbl4;
        java.awt.Label labelimcbl5;
        java.awt.Label labelimcbl6;
        java.awt.Label labelimcbl3;
        java.awt.Label labelimcbl7;
        java.awt.Label labelimcb6;
        java.awt.Label labelimcacad;
        symantec.itools.multimedia.ImageViewer imageViewer3;
        symantec.itools.awt.InvisibleButton invisibleButtonb3;
        symantec.itools.awt.InvisibleButton invisibleButtonlib;
        symantec.itools.awt.InvisibleButton invisibleButtonguild;
        symantec.itools.awt.InvisibleButton invisibleButtoncan;
        symantec.itools.awt.InvisibleButton invisibleButtonbl4;
        symantec.itools.awt.InvisibleButton invisibleButtonbl3;
        symantec.itools.awt.InvisibleButton invisibleButtonbl5;
        symantec.itools.awt.InvisibleButton invisibleButtonbl6;
        symantec.itools.awt.InvisibleButton invisibleButtonbl7;
        symantec.itools.awt.InvisibleButton invisibleButtonb6;
        symantec.itools.awt.InvisibleButton invisibleButtonacad;
        java.awt.Label label5;
        java.awt.Label label6;
        java.awt.Label label7;
        java.awt.TextField textField4;
        java.awt.TextField textField5;
        symantec.itools.multimedia.ImageViewer reddot2;
        java.awt.TextField textField3;
        symantec.itools.awt.BorderPanel ts;
        symantec.itools.multimedia.ImageViewer imageViewer5;
        java.awt.TextField tstext;
        symantec.itools.multimedia.ImageViewer libraryhigh;
        symantec.itools.multimedia.ImageViewer canteenhigh;
        symantec.itools.multimedia.ImageViewer guildhigh;
        symantec.itools.multimedia.ImageViewer b3high;
        symantec.itools.multimedia.ImageViewer bl3high;
        symantec.itools.multimedia.ImageViewer bl4high;
        symantec.itools.multimedia.ImageViewer bl5high;
        symantec.itools.multimedia.ImageViewer bl6high;
        symantec.itools.multimedia.ImageViewer bl7high;
        symantec.itools.multimedia.ImageViewer b6high;
        symantec.itools.multimedia.ImageViewer bl8high;
        symantec.itools.multimedia.ImageViewer reddot3;
        symantec.itools.awt.BorderPanel env;
        java.awt.Button button5;
        java.awt.Button button12;
        java.awt.TextField textField6;
        java.awt.Label label15;
        java.awt.Label label16;
        java.awt.Label label17;
        java.awt.Button button1;
        java.awt.Label label18;
        java.awt.Label label19;
        java.awt.Label label20;
        java.awt.Label label21;
        java.awt.Label label22;
        symantec.itools.multimedia.ImageViewer eastp;
        symantec.itools.multimedia.ImageViewer northp;
        symantec.itools.multimedia.ImageViewer southp;
        symantec.itools.multimedia.ImageViewer westp;
        //}}
}



//HELP.java
/*
    help frame for the applet
 */
```

```
import java.awt.*;
import java.net.*;
import java.io.*;
import java.applet.*;

public class HELP extends Frame {

    URL fileURL;
    InputStream input;
    DataInputStream dataInput;

        void exit_MouseDown(Event event) {
                hide();
        }
        public HELP() {

                //{{INIT_CONTROLS
                setLayout(null);
                addNotify();
                resize(insets().left + insets().right + 475,insets().top +
insets().bottom + 427);
                imageViewer1 = new symantec.itools.multimedia.ImageViewer();
                imageViewer1.reshape(insets().left + 0,insets().top + 0,474,428);
                add(imageViewer1);
                try {
                        imageViewer1.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/helpback.gif"));
                } catch (java.net.MalformedURLException error) {
                }
                env = new symantec.itools.awt.InvisibleButton();
                env.reshape(insets().left + 12,insets().top + 48,120,20);
                add(env);
                ts = new symantec.itools.awt.InvisibleButton();
                ts.reshape(insets().left + 12,insets().top + 96,126,18);
                add(ts);
                imagemap = new symantec.itools.awt.InvisibleButton();
                imagemap.reshape(insets().left + 24,insets().top + 132,108,24);
                add(imagemap);
                pathsmaps = new symantec.itools.awt.InvisibleButton();
                pathsmaps.reshape(insets().left + 12,insets().top + 180,128,28);
                add(pathsmaps);
                userinput = new symantec.itools.awt.InvisibleButton();
                userinput.reshape(insets().left + 12,insets().top + 216,110,27);
                add(userinput);
                tele = new symantec.itools.awt.InvisibleButton();
                tele.reshape(insets().left + 0,insets().top + 264,131,21);
                add(tele);
                pf = new symantec.itools.awt.InvisibleButton();
                pf.reshape(insets().left + 12,insets().top + 300,112,22);
                add(pf);
                exit = new symantec.itools.awt.InvisibleButton();
                exit.reshape(insets().left + 36,insets().top + 372,48,25);
                add(exit);
                try {
                textArea1 = new java.awt.TextArea(60,0);
                fileURL = new URL("help.txt");
                }
                catch (MalformedURLException e) {
                    textArea1.appendText("gone wrong");
                }
                textArea1.reshape(insets().left + 168,insets().top + 24,295,375);
                add(textArea1);
                setTitle("HELP");
                //}}
        }

        public void start ()
        {
            String text;

            try{
                input = fileURL.openStream();
                dataInput = new DataInputStream(input);
                while ((text = dataInput.readLine()) !=null)
                textArea1.appendText(text + "\n");
                dataInput.close();
            }
            catch(IOException e)
            {textArea1.appendText("second gone wrong");}
        }
        public HELP(String title) {
            this();
            setTitle(title);
        }

    public synchronized void show() {
        move(50, 50);
        super.show();
    }

        public boolean handleEvent(Event event) {
        if (event.id == Event.WINDOW_DESTROY) {
            hide();
            return true;
        }
                if (event.target == exit && event.id == Event.MOUSE_DOWN) {
                        exit_MouseDown(event);
                        return true;
                }
                return super.handleEvent(event);
        }

        //{{DECLARE_CONTROLS
        symantec.itools.multimedia.ImageViewer imageViewer1;
        symantec.itools.awt.InvisibleButton env;
        symantec.itools.awt.InvisibleButton ts;
        symantec.itools.awt.InvisibleButton imagemap;
        symantec.itools.awt.InvisibleButton pathsmaps;
        symantec.itools.awt.InvisibleButton userinput;
        symantec.itools.awt.InvisibleButton tele;
        symantec.itools.awt.InvisibleButton pf;
        symantec.itools.awt.InvisibleButton exit;
        java.awt.TextArea textArea1;
        //}}
}




//CREDIT.java
/*
    credits frame for the applet
 */

import java.awt.*;

public class CREDITS extends Frame{

        public CREDITS() {

                //{{INIT_CONTROLS
                setLayout(null);
                addNotify();
                resize(insets().left + insets().right + 430,insets().top +
insets().bottom + 270);
                setFont(new Font("Dialog", Font.BOLD, 12));
                setForeground(new Color(0));
                setBackground(new Color(16777215));
                credit = new symantec.itools.multimedia.ImageViewer();
                credit.reshape(insets().left + 0,insets().top + 0,430,270);
                add(credit);
                try {
                        credit.setURL(new
java.net.URL("file:/C:/WINDOWS/Desktop/PROJECT/html/credit.GIF"));
                } catch (java.net.MalformedURLException error) {
                }
                setTitle("CREDITS");
                //}}
        }

    public synchronized void show() {
        move(50, 50);
        super.show();
    }

        public boolean handleEvent(Event event) {
        if (event.id == Event.WINDOW_DESTROY) {
            hide();
            return true;
        }
               return super.handleEvent(event);
        }
        //{{DECLARE_CONTROLS
        symantec.itools.multimedia.ImageViewer credit;
        //}}
}
```