Web-Based 3D Environment for Urban Houses

by

MUHAFIZA BINTI MUSA

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information Communication Technology)

JULY 2005

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t
T385
.M952
2005
1) Computer graphic
2) IT/IS -- Thesis

CERTIFICATION OF APPROVAL

Web-Based 3D Environment for Urban Houses

By

Muhafiza binti Hj. Musa

A **project dissertation** submitted to the
Information Technology Program
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
Bachelor of Technology (Hons)
Information Communication Technology

Approved by,

[illegible]

(Dr Wan Fatimah Wan Ahmad)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
July 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

(MUHAFIZA BINTI MUSA)

ABSTRACT

After the emergence of technologies especially in Information, Communication and Technology (ICT) industry, many construction companies have ventured in the e-business in order to be more competitive. One of the interactive multimedia tools such as virtual reality (VR) have the potential to enhance the efficiency and effectiveness of all stages of a project, from initial conceptual design through detailed design, planning and preparation, to construction completion. The objective of this project is to develop a prototype which run on web-based that totally helps the customers to visualize the real house they are looking for by experiencing, in near-reality sense of unlimited virtual walkthrough. In the meantime, it will less the cost of money for transportation as well as the time taken for the customers to the advertised properties location. For the construction side the facility in a 3D interactive and immersive environment can increase the understanding of the design intent, improve the constructability of the project, and minimize changes and abortive work that can be detected prior to the start of construction. Basically, nowadays customers need to a site visit in order to see the house that they are interested to buy. Other than that, most of them are too busy due to their working hours and some of them live far away from the advertised properties location. By the emergence of the computer graphics, software technologies, internet and interactive multimedia tools, many developer companies have ventured in e-business in order to solve the problem that counter by the customers. The scope of study of this project will focus on how VR could help the construction company by creative an immersive environment for their show houses. For this project, System Development Life Cycle (SDLC) has been chosen because it provides systematic and orderly approach in solving system problem. The findings from the project will be determining base on the client satisfaction and evaluation toward the product later on.

ACKNOWLEDGEMENT

In the Name of Allah The Most Gracious and The Most Merciful. Peace and blessing be upon our Grate Prophet Muhammad Sallahu 'Alaihi Wassalam.

I am indebted to many individuals who contributed in various ways in completing my project dissertation successfully. I wish to extend my greatest appreciation for precious time, assistants, guidance and support offered to me by the following parties: -

Dr Wan Fatimah Wan Ahmad (FYP Supervisor)
Mohd Zuhairi Mohd Nor (Project Engineer OSK Properties)
Mr. Nordin Zakaria (FYP Coordinator)
Hj Musa bin Awang & Hjh. Hasnah binti Shafie (beloved parents)
All staffs in OSK Properties Sg. Petani, Kedah
All staffs in IT/IS Department
All my friends

Not to forget to those who involved directly or indirectly for ensuring the timely completion of this report. My final draft report would not have been possible and success as it now without your help and support.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. BACKGROUND OF STUDY

The usability of Web sites is becoming a very important topic as organizations struggle to reach, and especially to retain, a wider audience. It describes the many business, technical and other benefits to the organization above and beyond the straightforward benefits to people with disabilities that can be realized by applying the Virtual Environment (VE) to Web sites. Virtual Reality (VR) walkthrough as a branch to VR has been used by many sectors and fields of work to visualize and represent their work in a more elaborate way. By incorporating a VR walkthrough on online, it will reflect its advancement in technology as whole as well as the competitive advantage between other companies.

1.2. PROBLEM STATEMENT

1.2.1. Problem Identification

The main problem in this project is obviously to create a multimedia-advertising module that look realistic as well as capable to run on online basis. Nowadays, customers need to do site visit in order to see the particular house that they are interested to buy. Some of the customers have less of time and some of them are living far away from the advertised properties' location. In the meantime, the developers are confronting a competitive edge between their rivals on how they can get the biggest profit to their company through advertising. This kind of thinking actually has driven the properties developers to use the advertisement sector in order to attract the buyers.

As an initiation, the website could provide a virtual walkthrough of the house properties. This could give an advantage to the company to attract more and potentials buyers as oppose to the other company that lack such as promotions.

1.2.2. Significance of the Project

Internet and intranet web sites have become an integral part of software development as well as the wider fields of commercial, educational and recreational activity. In order to improve the understanding of web based software engineering it's required to understand first how web sites are used. Information about the structure and usage of sites is valuable to administrators, maintainers, content developers and users. The volume and complexity of the data generated by typical tools is a major limitation. Its describe the use of virtual worlds, including a novel extension of the perspective wall, for visualizing web site activity. Examples from the environment are presented and discussed. A website can benefit almost any business. While it can be a very effective marketing tool it is more important as a business essential – since the customers expect it. By incorporating VR walkthrough in the website, the customers will have the chance to see and sense the houses in term of design and its dimension. Comparing to merely pictures and textual description, a VR walkthrough would give better feel of how the houses would be look like physically. A walkthrough will give a sense of immersiveness to the user where the users feel as though he or she is in the actual environment itself.

1.3. OBJECTIVES AND SCOPE OF STUDY

1.3.1. Objectives

1. To develop a prototype which runs on web-based that totally helps the customers to visualize the real house they are looking for by experiencing, in near-reality sense of unlimited virtual walkthrough

2. To minimize the cost of money for transportation as well as the time taken for the customers to the advertised properties location
3. To give more controls to customers by giving them a range of choices in their experience with product information
4. To present the immersiveness of VE in the product as which users feel interactive, realistic with and within which people can interact

1.3.2. Scope of Study

The scope of this project is about developing a VR walkthrough on a website. The main idea was to promote the new residential area that has just been open in Sungai Petani. Collaboration has been done with the construction company, the OSK Properties Sdn. Bhd. in order to make the project more successful. For this project, several houses have been chosen to be developed such as Iris II (Single Storey Semi-D Bungalows) and Casa (Single Storey Terrace House).This urban houses will be developed in 3D environment which allow user to interact and view freely on the website.

1.3.3. The Relevancy of the Project

By the end of timeline, the projects mainly will covers on research and solution the problems incorporating of findings into the prototype, multimedia authoring as well as the application principles and techniques. To complete this project, there are two approaches will be taken such as:

- Research on VR that shown how it was implemented in the market nowadays. Research is done from various organization backgrounds which shown that VR is built from different types of styles and for certain purposes.
- Research continues on how the immersiveness in VE has helped marketing area to produce better wealth and profits from their products.
- Nevertheless, research is also covering on website purposes, which function as the platform to show the virtual products to customer that came from all over the world.

1.3.4 Feasibility of the Project within Scope and Time Frame

By the end of the semester, the product should be completed as a VR walkthrough in the website. A short montage will be put as the opening of the website. The montage will be done depending to the time consuming. If there left anytime till the presentation, probably the chance to develop it is higher. Later then, is to upload the VR walkthrough into the website. This is the most critical time during the project since the development stage only has taken half of the project time frame. To make the navigation successful, a plug-ins should be downloading before the user want to view the VE. However, limiting the final product to be only one VE, would buy enough time to enhance it and at the meantime developing the website as well.

CHAPTER 2
LITERATURE REVIEW AND THEORY

Traditional methods in interior design usually lack depth and sense of realism, as well as require the designer and the client to meet in one place. These problems can be solved by utilizing shared virtual reality in the design process and embedded it on websites. Information can be provided on your site, not only reducing the printing costs but also fax and mailing costs. Using the services, the customer doesn't need to travel all the way from their home. The proposed services can be used to greatly enhance the feeling of presence and help the physically challenged learn to cope with their environment

2.1 What is Virtual Reality?

There are many definitions of VR. According to physicist Deutsch (2000), VR is the best physical demonstration of universality, one of the most important concepts of the theory of the calculability (the own difficulty of the calculation processes and its limitations). The best and clearer physical manifestation of this theory is the virtual reality, which the author defines as "any situation in which a person goes, in an artificial way, through the experience of being in a specific environment." To develop a usable VR system, the prospective context of use of such a system may need to be considered in order to make sure it meets the requirements and restrictions of that context

However for the project purposes, the most useful definition is the final product comparative level of performance in reaching its objectives. This implies of having experiences where the software learned which action is the best helps it reach its objectives. VR provides multisensory environments for learners to interact in a context. Gibson's ecological psychology (1986) recommends VR's active and integrated perceptualizations, not merely visualizations for learners to create, observe, and understand relationships within the environment. Despite focusing on visualization, learners or creator also learn through the perceptualizations in the context of human presence, touch and feels in the immersive environment. . The identified potential usability problems of a fully immersive prototype, coupled with the needs, requirements and real-life environment of the end-users lead to guidelines for the development of a VR application on a semi-immersive desktop environment. The findings lead us to believe that contextual analysis can be a powerful way to inform the design of a VR application by offering an understanding of the context of use and to inform developers of the most appropriate degree of immersiveness of the VR environment

2.2 The immersiveness in VR

Virtual reality (VR) applications are often developed relatively independent from the real contexts in which they are going to be used. However, it is recognized that user needs should play a central role in the development of virtual environments that are to be used in a real-life context, an insight that has existed in the systems development community for years. Three categories of VR often used are desktop, semi-immersive and fully immersive VR. Many authors use to classify VR environment on the basis of the level of immersivity. This can be considered a quality measurement of the presence feeling or, similarly, the perception of the virtual world as a real world.

- **Desktop VR -** Called Desktop VR or a Window on a World (WoW), this type of VR yields a standard monitor and common keyboard and mouse to interact with virtual world. Not very far from a modern CAD system, where object and data can be managed and visualized in 3D fashion, is the quality of the final simulation quite poor due to a limited operator's FOV (Field Of View) and a 2D interface. Furthermore the small display dimensions also reduce the need of a head tracking, which is the collimation of the virtual camera and operator's head. Instead a great benefit is provided by the very low cost, because the no special hardware is required

- **Semi-immersive VR** - Time and cost reduction has been a critical success factor for the rapid development of VR semi-immersive systems. Reproducing in the virtual environment the whole real environment surrounding the user in a perfect reconstruction process results, in fact, extremely time consuming. A high calculation power is requested, for example, to reproduce the user's hand in the virtual environment due to the huge number of polygons and to the kinematics constrains. Furthermore, one of the most appreciated features in these systems is the multi-user capability. In a project verification step and much more in a final project approval it is particularly needed a presentation device, which is able to make users navigate inside and around the photorealistic under-approval product.

- **Full Immersive VR-** Full-immersive visual systems represent the higher level of user immersiveness in virtual environments. It can be provided by an HMD (Head Mounted Display) or by a binocular oriental monitor (such as Fakespace BOOM), both reproducing the Desktop VR, Fakespace BOOM and Head Mounted Display.

By referring to Cramer, Evers, Zudilova and Sloot, it proof that desktop VR can be offered as either immersive with, for example, shutter glasses, or non-immersive without specialized equipment. Currently, data on the appropriate degree of immersion for a given situation is scarce. There is no complete framework available facilitating the choice whether a (non-VR) desktop application or VR, fish tank, semi-immersive or completely immersive virtual reality application might be the best choice for a particular system and context of use. The choice between these options can only be made after considerations of usability criteria for an application and analysis of the context in which a system will be used.

2.3 VR in Website Visualization

Few major commercial or educational enterprises would expect to conduct their business effectively without an Internet presence. Typically one or more web sites are used for internal or external purposes. Increasingly, software engineering takes place in an Internet/Intranet environment. Software artifacts as well as processes are developed, delivered and deployed using web-based technology. User manuals and technical documentation are routinely supplied as HyperText Mark-up Language (HTML) format—often with the intention of being accessed remotely rather than being downloaded *in toto* by users. Effective re-use depends on the ability to locate suitable components and their APIs. The Java development kit includes a tool, javadoc, which generates API documentation in HTML format and similar tools have been in use for some time.

Literate programming tools provide yet another source of web-based software artifacts and one of the ways that source code is also being used in a web-based form. It is important for software engineers to be able to comprehend measure and manage web-based systems as well as those of a more traditional development environment. As with other aspects of software engineering, the size and complexity of web sites are challenging to conventional techniques. In this paper we describe some of our applications of information visualization techniques to web site log data. Similar ideas have been applied in other areas of software engineering .A better understanding of web site usage can then be applied to issues specific to software engineering as well as those of more general applicability.

Web server software varies considerably but individual products are capable of tracking activity in the form of log files of various kinds. Such files are typically rather too large (40MB per week in our case) for ready comprehension by human readers. File and version management involves maintaining the file system structure, dealing with updated versions of individual resources and the addition of new material. Hyperlink management includes the removal of "dead" links and the creation of new links in response to observed user demand. Although the power and sophistication of the tools available to assist site maintainers is increasing, it is still difficult to manage both fine detail and large scale information. There are many reasons why information about the structure of a site and the pattern of visitor behavior is useful. These range from site design to marketing banner advertising. According to Hartley and Churcher,

VR has advanced from the realm of fiction and has been applied in many different domains. Both the general aspects of VR as well as more technical issues have been described extensively.

By looking back to the statement above, a conclusion can be made that in order to achieve the aim of delivering useful visualizations on standard platforms, it actually limit to non-immersive VR; achieved by using an ordinary display screen to give the impression of navigating through a 3-dimensional space and will not be concerned with the data gloves, helmets and specialized systems found in immersive VR environments. By using the Virtual Reality Modeling Language (VRML) or other optional software to represent the results. VRML is essentially a scene description language and is more suitable for our purposes than libraries such as java3d which extend general purpose programming languages.

2.4 VR – The Benefits

The big area that is gaining momentum is the internet. VR can be added to its interface to make the net a true "cyberspace." By adding the capability of adding 3D interactive graphics to a web page, the web revolution can maintain its momentum. This was all made possible with the invention of VRML (visual reality markup language). VRML along with java allow entire 3D interactive worlds to be created from a single web page. Although not widely used today, the technology also allows for shared virtual environments where someday you may be able to interact with other people from across the world in a virtual world from a central web site.

VR technologies address a wide range of interaction and immersion capabilities during the VR experience. Immersion varies from first, second-, or third-person experiences and in physical, perceptual, and psychological options.
According to Mahoney (1994),

> *"Architects and designers are increasing their use of computer-generated walkthroughs of their designs, and while the various types of walkthrough, or flythrough, differ, they all involve motion"*

By referring to the quotes above, it is proved that VR can engage learners at the emotional level through the "willing suspension of disbelief". Its immersion characteristic utilizes role-playing and varying perspectives as a fundamental way of knowing. Virtual "agents" or human-like avatars can be the dramatic characters if instructional designers desire more predictability and control than possible with human models.

2.5 Main Components and Devices

According to Wicken and Baker (1994), said that VR has five main components which are variable according per the instructional context requirements:

- dimensionality,
- motion or animation,
- interaction,
- viewpoint or frame of reference, and
- Immersion, or embodiment, through enhanced multisensory experiences.

By referring to Wicken and Baker (1994) above, some instructional contexts depend on specific realities, which the contexts that never vary. Scientists seeking medical cures must observe atomic particles in specific places and with specific behaviors when manipulated. In the other hand, some instructional contexts require nonspecific realities; which the contexts with naturally varying conditions. VR's range of realities enables instructional designers and practioners to increase contextual complexities as learners' transition from novices to experts.

Persiani (2001) in her article wrote that, VR is a way for humans to visualize, manipulate and interact with computers and extremely complex data. A good definition to point the attention on the TWO main VR behaviors: enhanced visualization and interaction. Both of them mean real-time elaboration. All VR environments can mix in different levels these items and the most part of the rest of this paper is intended to clarify the role of these features to allow the operator to view and interact with data in a 3D environment, where reality may be reconstructed or completely simulated. Following leading features already cited (enhanced visualization and interaction), starts to describe the most important hardware equipment used by VR.

A general VR application, wishing to provide to the operator a 3D feeling of the surrounding virtual world, usually exploits the capabilities of a **stereo projection**. The stereo vision is performed by active stereography or passive stereography. The first is most largely used due to a better 3D experience: this is accomplished by creating two different images of the world, one for each eye. The images are computed with the viewpoints offset by the equivalent distance between the eyes. The two images can be displayed sequentially on conventional monitor or projection display. **Liquid Crystal** shutter glasses are then used to shut off alternate eyes in synchronization with the display. When the brain receives the images in rapid enough succession, it fuses the images into a single scene and perceives depth. A fairly high display swapping rate (min. 60 Hz) is required to avoid perceived flicker.

Passive stereoscopy instead is used in very low cost applications and is based on the production of the images through differently polarized filters, with corresponding filters placed in front of the eyes. Anaglyph images use red or blue glasses to provide a crude (no color) stereovision. On the other hand, the Input Processes of a VR program control the devices used to input information to the computer (**motion tracking**). There are a wide variety of possible input devices: keyboard, mouse, trackball, joystick, 3D & 6D position trackers (glove, wand, head tracker, body suit, etc.). All these devices return a relative spatial position with respect a static receiver, while several sensors are placed on operator's hands and head. **Tracking devices** handle the interactions, the scripted object actions, simulations of physical laws (real or imaginary) and determines the world status. This simulation is basically a discrete process that is iterated once for each time step or frame. It is the simulation engine that takes the user inputs along with any tasks programmed into the world such as collision detection, scripts, etc. and determines the actions that will take place in the virtual world.

Above was some example of input and output devices used in VR. The application run from VR usually will be connected to these types of devices. By looking back to the project, user doesn't need any output devices in order to explore the walkthrough. The main requirement was to download the plug-ins in order to view the VE provided in the sites.

CHAPTER 3
METHODOLOGY

This chapter will cover on how the project being conducted and done. The first part will show about the flow of the project starting from the interface. The second part will focus on procedure identification of the project. On this section it will cover on methodology used in the project. Follow by is the tools used which contained of software and hardware requirement of the project. Lastly, will be the design section which covers all about the software used in the project.

3.1. PROJECT FLOW

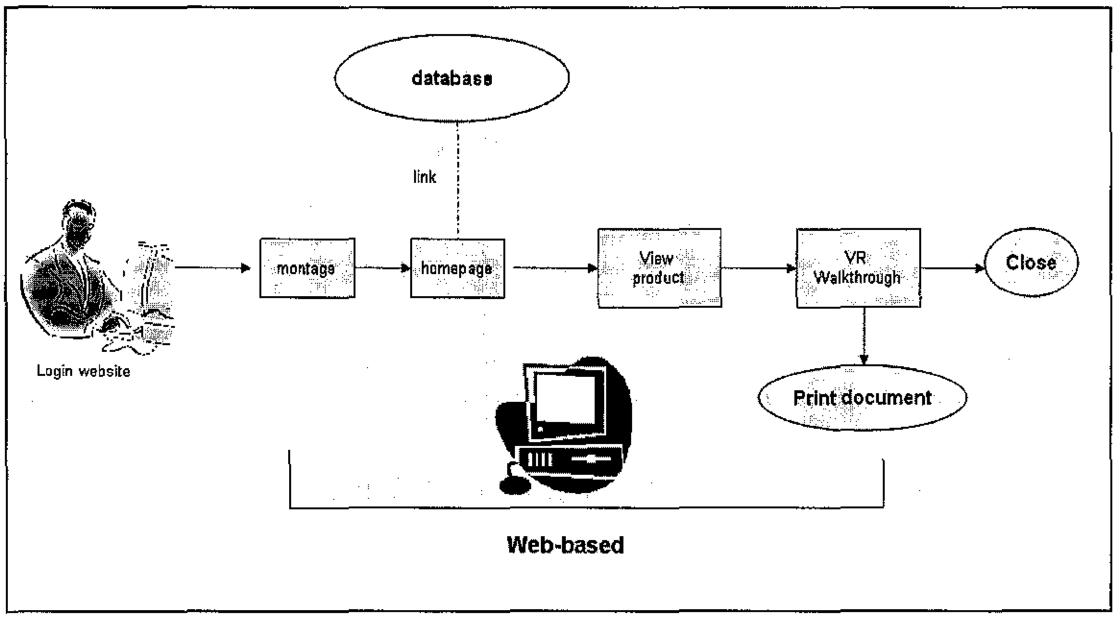


Figure 3.0: Process Flow

From the figure 3.0 above, it explains about on how the process flow happens in the project. The main purpose of the figure above is to show how the user will use the website in order to browse the advertised houses. The flow starts when the user begins to navigate the webpage. The opening will be a short montage that brief about the company latest project. User can skip the montage in order to reach the main page faster. Starting at the main page, the search index has the connectivity with mySql database. From the main page; the user can navigate to other pages in order to view all the products. Several of the promoted products are given a VR Walkthrough in their pages (refer to the figure 4.13) which show the linked button.Lastly, the user can close the site or print out the document they needed.

3.2 PROCEDURE IDENTIFICATION

Throughout this chapter, the systematic approach analysts take to the analyst and design of information. Much of this embodied in what is called the systems development life cycle (SDLC). The SDLC is a phased approach to analysis and design that holds that systems are best developed through the use of a specific cycle of analyst and user activities.

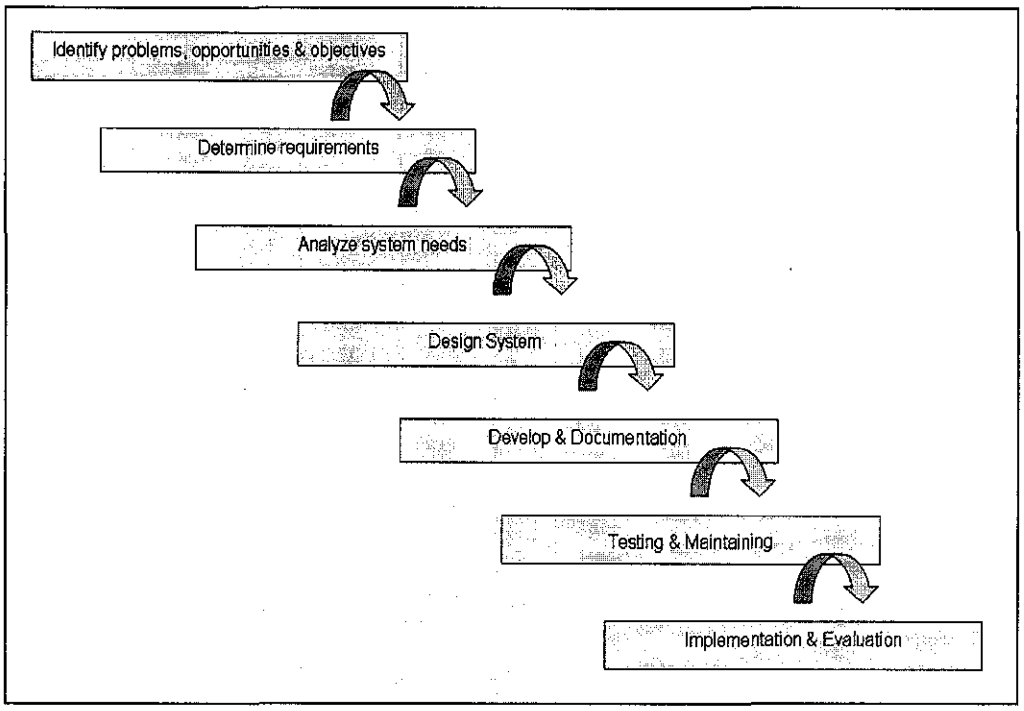


Figure 3.1: Software Development Life Cycle

This life cycle can be divided into seven sequential phases, although in reality that phases are interrelated and are often accomplished simultaneously. The seven phases are identifying problems, opportunities, and objectives; determining information requirements; analyzing system needs; designing the recommend system; developing and documenting software, testing and maintaining and lastly implementing and evaluating final products.

- **Identifying problems, opportunities, and objectives**

  The first stage is to define problems of the projects that being faced by target user. When the problems have been identified, and then know what are the opportunities and objectives to be achieved through the projects.

- **Determine information requirements**

  This section involves the research part, which required finding of information about the projects requirement. Research can be done either through articles, journals from library or surfing the net. Rather than that, information about clients or user is the most important of all. Designers need to know what the clients want before start the developing stage.

- **Analyzing system needs**

  Since the project requires a walkthrough that can be view from the net, a lot of research must be done in order to understand what the system need. The research will cater on website requirements, software chosen and also the hardware used through the projects

- **Designing the recommend system**

  When all the inputs or data ready, the next stage is to design the flow of recommend system. From these sketches, designer will see thoroughly the whole system before pass it to the developers.

- **Developing and documenting software**

  In this stage, the most critical time of all required me to develop the prototype. At the meantime, all the research before must be documented as a report as well as a reference to the reader. All the information gathered must be kept and arranged nicely in a hardcover report (dissertation).

- **Testing and maintaining**

  When the product has been finished developed, it will be test in order to see is capability. Usually, failures always occur in this stage and quickly it will be repair as soon as possible. Some enhancement and improvement will be done to the product before production. Maintaining will be the last past in order to test the strength capability and quality of the products.

- **Implementing and evaluating final products**

  The final stage is implementation into real environment. The products will be present to the management first in order to see the response. Later, when the management satisfies with the products, it will be present to the customers. Evaluation will be start soon as the products being received by customer. It's very important to know the customers reaction and response for further action and enhancement.

3.3 TOOLS

The following are the tools suggested to be used in the development of the project:-

Hardware Requirement

- PC running on Windows 95/98/2000/ME/NT/XP (i386)
- High Performance Graphic Card (ATI RADEON 9250)
- Intel Pentium 4 with processor of 1.6 GHz, Ram of 512 mb
- Digicam Sony 3.0 pixels

Software Requirement

- Adobe Atmosphere 1.0
- Adobe Photoshop 7.0
- Adobe Acrobat Reader 7.0
- Macromedia Director Mx
- Web Publisher (Macromedia Dreamweaver)

There was a change of software during the developing stage. The main software of Blender 3D, has been changed to Adobe Atmosphere 1.0 before the development stage begun. Between Blender and Adobe, it is clear that Adobe is more user friendly and easier to use comparing the VR walkthrough produced by Blender. Therefore, this software has chosen the Adobe as the platforms to develop the VE in this project.

3.4 DESIGN

3.4.1 Use Case Diagram

The use case diagram describes the interaction between user and the application. As illustrated in the figure 3.2 below, the main interaction between user and the website is the VR Walkthrough. Firstly, user need to visit their interested product and just click to the button provided at the bottom of the page. From there, it will link you to another page which will display the VR directly. The user does not need to install plug-in like other software, which this kind application is friendlier user.

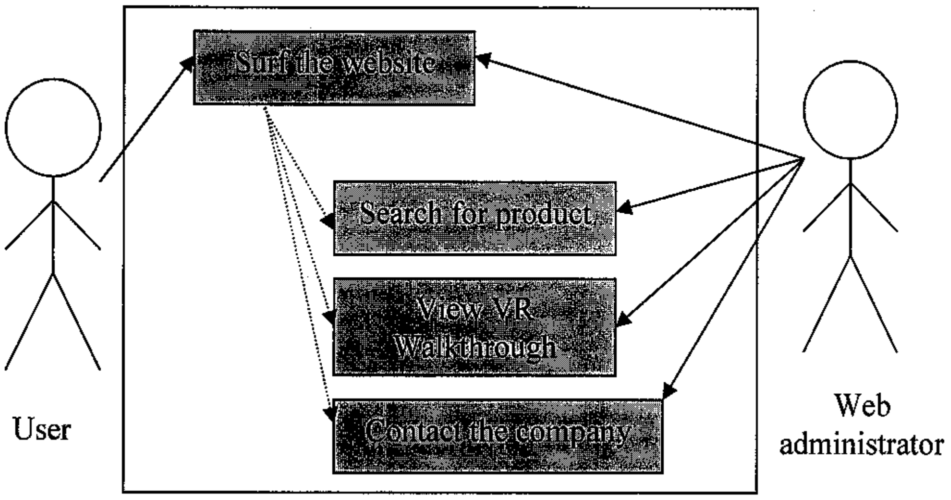


Figure 3.2: Use Case Diagram

3.4.2 Storyboard

The figure provided below (figure 3.3) will represent as all the basic interface of the website and the arrangement of the content display on it.

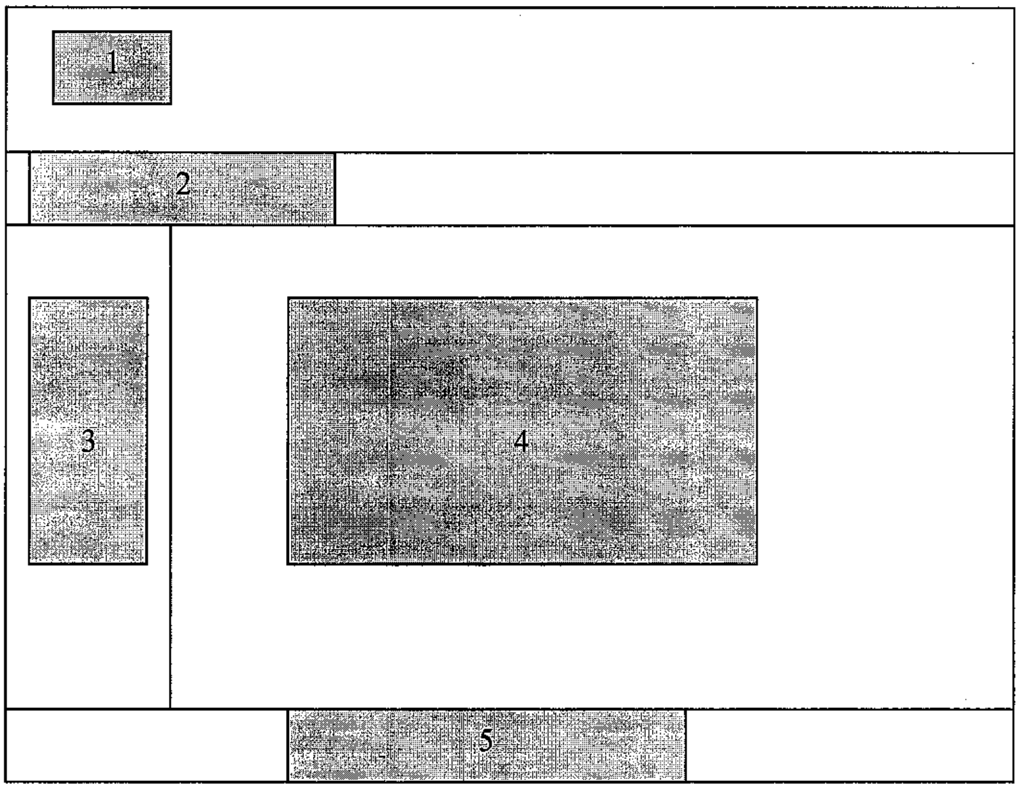


Figure 3.3: Storyboard

1: the company's name and logo
2: the buttons, which contain of homepage, products, prices, FAQ, about us and etc. which will link to the appropriate pages
3: provide company news and updated product. This side also providing link directly to the news or houses posted.
4: page content
5: important buttons as stated above, be repeated again at the bottom

3.5 DEVELOPMENT

3.5.1 The Atmosphere Platform

- **Platform Features**

  The Atmosphere platform integrates many features into a single application:

  - **Immersion** Create environments with dramatic lighting, animated 3-D objects, real-time behavior, video and audio in small and less embedded in a PDF or HTML document.
  - **Interactivity** Atmosphere users can easily create realistic environments and behavior thanks to the built-in JavaScript API and integrated physics engine. Users can interact realistically with environments, objects and avatars that can behave independently or under user control.
  - **Multimedia** Atmosphere supports directional sounds, streaming video and audio, SWF animations and high definition 3-D objects including animations.
  - **Collaboration** Add multi-user interaction with text chat, avatar motion and gestures, shared object synchronization and message passing, without additional server software or hardware. An HTTP server is all that's required.

- **The Atmosphere Architecture**

  The Atmosphere platform consists of 3 parts:

  - **Atmosphere** An easy to use, powerful authoring application that can be used to create, import and manipulate 3-D objects, light, sound, images, textures, video and other multimedia; attach scripted behaviors to objects and environments; and securely publish these environments to the Web or PDF.
  - **Atmosphere Player** A web browser plug-in that allows users to interactively view and navigate Atmosphere environments embedded in web pages. Atmosphere Player for Adobe Reader™ enables users to access Atmosphere environments embedded in PDF documents.

- **Atmosphere Collaboration Server** A publicly available server, which allows messaging, object synchronization and interactivity between users of Atmosphere environments.

- **Atmosphere Player** The Adobe® Atmosphere™ Player® is a free application that allows users to view, interact, and collaborate with others in environments created with Atmosphere and published online or within PDF documents. Atmosphere environments can be viewed within a web page using the Player plug-in.

- **Web Page Integration** Atmosphere Player supports communication between web page HTML, Java and JavaScript allowing web designers full control of user experience using the Player's JavaScript API.

- **A Multimedia Experience** The Atmosphere Player offers the user a rich and interactive multimedia experience without requiring a high-bandwidth connection (56K normally suffices). Atmosphere environments can also be viewed in

- **Atmosphere Player System Requirements:**
  - Intel Pentium II or faster processor
  - Microsoft® Windows® 98SE, Windows ME, Windows 2000, Windows XP Home or Pro
  - 64 MB of available RAM (128 MB recommended)
  - 14 MB of available hard-disk space
  - 16-bit color (32-Bit Color recommended)
  - 56K modem or faster Internet connection
  - Microsoft Internet Explorer 5 and above.
  - Graphic Card Support: Radeon 7500 or higher, GeForce 2 or higher

3.5.2 Exploring Atmosphere Environment

Before we turn our attention to learning how to create interactive environments in Atmosphere, let's learn how to use the Atmosphere Player. The Player is a free plug-in that allows people to view and interact with Atmosphere environments embedded in web pages and PDF documents.

- **Using the Atmosphere Player**

When you installed Atmosphere, the Player was installed automatically. If you need help with Atmosphere installation, please see Appendix A, "Installing and Configuring Atmosphere." Once installed, Atmosphere environments will be loaded in the Player automatically when you navigate to a web page or open a PDF document that contains an Atmosphere. If the Player isn't installed, a dialog box will appear asking if you wish to install it. The Atmosphere Player features support for hardware acceleration. If the Player detects a supported video card, it will enable hardware acceleration automatically. This animated spinning globe icon and a progress bar will appear as the Atmosphere environment is loading. When the spinning globe icon disappears, the geometry for the environment has completed loading and will be displayed. Textures and multimedia files will continue to download.

- **Opening Local Atmosphere Files**

In addition to Atmosphere environments that are posted online, you can also open environments that are saved locally. To open a local environment, right-click in the Player window and select File > Open from the pop-up menu. This will open a dialog box where you can select an Atmosphere environment to open. *The Player can only open and view AER files, not ATMO files.*

- **Navigating Atmosphere Environments**

There are several ways to navigate in Atmosphere Player, using the mouse or keyboard. Before navigate an Atmosphere environment, the environment must be the focus of the web page. Try by give the environment focus by clicking on it.

Movement Mouse motion Keyboard:

Move forward Drag mouse forward Up arrow

Move backwards Drag mouse backwards Down arrow

Turn to left Drag mouse to left Left arrow

Turn to right Drag mouse to right Right arrow

Ascend upward Shift+drag mouse upward Shift+up arrow

Descend downward Shift+drag mouse downward Shift+down arrow

Tilt view up Ctrl+drag mouse upward Ctrl+up arrow

Tilt view down Ctrl+drag mouse downward Ctrl+down arrow

Strafe left Shift+Drag mouse to left Shift+Left arrow

Strafe right Shift+Drag mouse to right Shift+Right arrow

- **Using the Toolbar**

If look in the lower left corner of an Atmosphere environment, it shows a toolbar of icons. The Atmosphere Player toolbar includes several icon buttons that toggle properties and open palettes. It can control several settings using these icon buttons. If forget which button does what, hold the mouse cursor over the top of the button and a tool tip will appear with the button name. Turn this toolbar on and off using the Contextual menu – right-click to bring up the pop-up menu and choose toolbar to make the icons appear or disappear. Try to access all the toolbar commands from the Contextual menu, which is convenient if the toolbar is hidden.

- **Detecting Collisions**

As moving through Atmosphere environments, notice that it cannot move through solid objects such as walls and pillars. This is because the Collide option is enabled. The Collide button allows user to toggle this feature, enabling movement through solid objects when turned off. *If have trouble navigating a particular environment, user can disable the Collide button to make it easier to move around.*

- **Setting Gravity**

Another physical constraint in Atmosphere environments is gravity. If the Gravity toggle button is enabled while user are higher than the floor of the environment, then user, and avatar - if turned on, will descend until user come to rest on a solid object. It won't descend unless both the Gravity and Collide button are enabled. If there is no solid object underneath, then it will continue to descend until the entire environment is out of site. If the Gravity toggle button is disabled, then it will hang in the air after releasing the mouse. *If user ever find he/her falling off the edge of environment, user can right click on the environment and select*

- **Understanding and Using Avatars**

As moving around in public Atmosphere environments on web sites, user may see other characters moving about. These characters are called avatars and they provide a visual representation of visitors to the environment. Each visitor has her own avatar. To see the avatar, click on the Show My Avatar button in the Player toolbar. An environment with the default avatar visible. Click the Show My Avatar button on the toolbar to see the avatar.

User can load different avatars using the Avatar palette. Clicking the Avatar button in the Player toolbar will open the Avatar palette. The top of the Avatar palette includes a box that shows a snapshot of your Current Avatar, another box that holds an avatar that can send to others, and a series of snapshots labeled My Avatars that holds a library of avatars that can swap at any time. Avatars, like bookmarks, can be shared with other users in an environment during a Chat session. Interacting with an Atmosphere Environment, explains how to do this. To change avatar, simply select one from the library in My Avatars and drag it to the Current Avatar box. Atmosphere will load the new character and present it in the Atmosphere window. User can delete an avatar by selecting it in the My Avatars sections and pressing the Delete key. If right-click in the My Avatars section, a Contextual menu will let user change the thumbnail size to Small, Medium or Large. If delete one of the default avatars from the My Avatars section there is no way to recover it unless you reinstall Atmosphere.

User can change avatars easily to suit the whim. The Avatar panel holds several different characters that can be use. To change the character, follow these steps.

I. Click the Avatar button in the Player toolbar to open the Avatar palette.
II. From the My Avatars section, drag the new avatar that you wish to use to the Current Avatar box at the top of the panel.
III. Enable the Show My Avatar button in the Player toolbar to see the new avatar.

- **Setting Player Preferences**

The Preferences button in the Player toolbar will open a palette with three separate tabs – General, Display and Chat. Each of these tabs includes settings which change how the Player works.

I. **General Preferences** The General tab of the Preferences palette includes settings for the avatar Nickname and URL under the Personal Defaults heading. The Nickname will appear in the Chat palette along with any message that type when in Chat mode. It will also identify in the Users palette to other visitors in the environment. To load a new avatar, simply enter the avatar's URL into the Avatar URL field and the avatar will show up as the Current Avatar in the Avatar palette. From here drag it into the My Avatars section. There is also a Show My Avatar option to make the avatar visible in the Atmosphere Player, which works the same way as the Show My Avatar button in the toolbar.
II. **Display Preferences** The Display tab of the Preferences palette lists and lets user choose Preferred Rendered. Options include Hardware (Direct3D) and Software. If the computer system has a video card that supports hardware rendering, then it will see better performance and image quality by selecting the Hardware (Direct3D) option as the Preferred Rendered. Software rendering, although slower, will always work regardless of the system. Even if select the Hardware (Direct3D) option, hardware rendering may be unavailable if Atmosphere is unable to recognize your video card.

Appendix A, "Installing and Configuring Atmosphere," includes information on configuring Atmosphere to recognize video card as well as a list of supported video cards.

III. **Chat Preferences** The Chat tab of the Preferences palette includes options to Enable Chat, Enable Chat Logging, Print URLs of Shared Links and Enable Bookmarks. At times, user may wish to turn off some of these options to maintain privacy. Note that by right-clicking in an environment, a Chat item appears in the pop-down menu. User can control the behavior of the Chat window – whether it appears at the bottom of the environment window or as a detached, floating window– by toggling this option. The Chat tab of the Preferences panel includes options for controlling the Chat features.

3.5.3 Interacting Atmosphere Environment

Atmosphere environments are compelling experiences because they allow users to interact with them realistically. For example, users can interact with other visitors and with objects in the scene that have scripted behaviors attached to them.

- **Moving Through Portals**

Much like web pages can be linked to each other, Atmosphere environments can be connected using portals. A portal's location is denoted by a set of spinning red, green, and blue squares. When a user's avatar is moved into a portal, the linked environment will automatically load in place of the existing one. Portals offer a way to link to other Atmosphere environments. User can select to move back and forward between linked environments using the Navigation > Back and Forward right-click pop-up menu commands. *If a Portal's squares are laying f at, then it is not active. This could be due to a broken link or a server that is offline.*

- **Jumping Between Environments**

When building Atmosphere environments, user may find it easier to work with several smaller interconnected environments that are linked using Portals. A good example of this is the Museum environment on the Atmosphere product pages at Adobe.com.To practice jumping between different Atmosphere environments using Portals, follow these steps.

I. Visit the Atmosphere product pages at Adobe.com by typing http://www.adobe.com/products/atmosphere/ into a web browser.
II. Locate the Museum Atmosphere environment in the Showcase pages and enter the Atmosphere scene. Try http://www.adobe.com/products/atmosphere/showcase/showmuseum.html
III. Navigate about the scene until you locate a Portal and move into it.
IV. The linked Atmosphere environment is loaded into the Player.

Portals offer a way to link to other Atmosphere environments.

- **Understanding Entry Points**

When first enter an Atmosphere environment, user will arrive at a location called the Entry Point that is designated by the environment creator. From the Atmosphere Player interface, user can have avatar immediately jump back to this location using the Navigation > Reenter World command on the Contextual (right-click) menu.

- **Communicating and Collaborating with Other Visitors**

When visit Atmosphere environments online, it can enable a Chat palette that allows text conversations with other online visitors if the environment has been designed for collaboration. Open a Chat palette by clicking on the Chat button in the Player toolbar. The Chat palette can also be used to display information about the environment. The Chat palette displays the text of online messages as users communicate with one another. User can also place the Chat window at the bottom of the Atmosphere environment by right-clicking and selecting Chat from the Contextual menu. The Chat palette can also be opened at the bottom of the Atmosphere Player interface.

- **Interacting with the Environment**

In addition to Portals and Chat, user can also interact with some Atmosphere environments through scripting that the designer has included. Some common interactivity options are listed below.

- **Sounds and movies** Within the Atmosphere environments, user can position sounds near objects that will grow louder as a user approaches, and softer as they move away. User can also add background sounds to the entire scene. Movies can be added as textures to any surface.
- **Interacting with the Web Page** Atmosphere environments embedded within web pages can interact with items on the page using standard HTML form elements such as drop-down lists and buttons.
- **Interacting with Scripted Objects** Scripts can add interactive behavior to any Atmosphere object. For example, scripts allow users to click on objects to initiate behavior. Imagine that a builder and are taking clients through an Atmosphere representation of their new home. Scripted behaviors would allow them to change options like paint color or tile patterns by clicking on the walls. The scene would be updated as clients make and evaluate their choices, allowing immediate visual feedback.

3.5.4 Atmosphere Overview

Adobe Atmosphere is a professional authoring tool for assembling and creating 3-D interactive stage sets. This new embedded multimedia type gives the web or document designer the ability to present a rich variety of interactive content, including three-dimensional objects, sound, streaming audio and video, SWF animations, and physical behaviors, all within the context of a live theater performance. The Atmosphere platform includes Atmosphere – an easy to use but powerful authoring tool designed to enable the creation of Atmosphere content; Atmosphere Player, web browser and Adobe Reader plug-in for navigating and interacting with 3-D environments on the Web and in PDF documents; and Atmosphere Collaboration Server which enables user communication and object synchronization.

Atmosphere scenes are composed of multiple assets. These assets can come from a variety of places. They can be created directly in Atmosphere, loaded from a library of pre-built objects, imported from other 2D and 3D design applications, or created at run-time using the Atmosphere Player's JavaScript API (Application Programming Interface). Assets of an Atmosphere scene can contain geometric, appearance, animation, and interaction information. Some of the assets that make up an Atmosphere scene are:

I. **2D Text** Created at run-time
II. **2D Sprites** Loaded at run-time
III. **Props** 3D Objects which are created in an external design application and either imported into a scene using Atmosphere or loaded at run-time
IV. **Surface Objects** 3D objects which are either created by importing other formats into Atmosphere and converting to this native type or converted by Atmosphere, then exported and loaded at run-time
V. **Solid Objects** 3D objects which are created in Atmosphere and either placed into a scene by Atmosphere or exported and loaded at run-time
VI. **Script Objects** - Imported into a scene using Atmosphere This phase of the Atmosphere workflow includes asset creation (Solid Objects), asset import (Props, Surface Objects, Solid Objects, Script Objects), and asset import and conversion (Surface Objects).

- **Scene Modeling**

In this phase of the workflow, combine scene assets geometrically, modify their appearance, and specify the appearance of the scene as a whole. This is the process of building a spatially and visually coherent 3D environment or stage set. Geometric combination includes positioning 3D assets and possibly scaling them in relationship to each other. Appearance editing includes applying color and textures to 3D object surfaces, and editing imported surface properties. Textures can include local or streamed video content in popular formats. Finally, global illumination computation can be used to make all the assets in the scene work together visually in a more realistic way.

- **Lighting and Light Maps**

The Lighting Control palette allows user to light Atmosphere scenes in the Appearance Editor's Player View. Atmosphere uses two different lighting methods – pre-computed and dynamic lighting. Pre-computed lighting information is placed in Light Maps that are saved along with the environment's geometry information. Dynamic lighting is applied in real-time using scripts. Atmosphere offers sophisticated lighting called Radiosity: you can learn more about lighting in .

3.5.5 Publishing

Publishing an environment is one of the final steps to make environment visible online. The process of publishing will bundle all of the files that make up the world–geometry, texture, light maps, sound, video and scripts–into a single, remarkably compact AER file ready for a web page or PDF file. Once published, the entire world will be able to see you creation, but they won't be able to open and "borrow" any of its components.

- **Testing and Debugging Environments** - It would be a pity, after all hard work, if the Atmosphere environment didn't behave as planned. To make sure users won't be disappointed; it's a good idea to test the environment before release it.
- **Locating Errors** - Experienced Atmosphere developers are familiar with some of the bugs that are common to online 3-D worlds, and make a point of checking for them. Atmosphere is perhaps the easiest-to-use 3-D package ever, but 3-D worlds are complex places, and bugs have a way of sneaking up on the most careful developer.
- **Inactive Portals** - If a Portal references a URL that cannot be located, then the normally spinning red, green and blue squares will lay flat. Flat Portal panels mean user should double check the URL listed for the Portal.

- **Leaking Light** - If create a room and the walls don't quite meet or align correctly with the ceiling, then light from the sun or background may leak in causing unwanted lighting in the scene. This is easy to check in the Player view. If don't need it for anything else, turn off the Sun's light and shadows in the Settings tab of the Lighting Control palette before lighting the environment.
- **Missing Textures** - If user imported Viewpoint Objects and converted them to Surface Objects that now seem to be missing their textures, then it may have forgotten to move the texture map saved as a PNG file to the server. The Publishing process will move textures applied to Solid Objects, but textures from converted Viewpoint Objects need to be moved by hand after being converted.
- **Scripting Errors** - If the environment generates scripting errors, they will be displayed in the Chat palette when the world is loaded in the Player. These errors will give the line number of the script and an error message that are helpful for debugging.
- **Incorrectly positioned Entry Point** - If the environment loads, but the floor seems to be missing, the problem could be that Entry Point object is contained within the Floor object. Another problem occurs when Entry Points are positioned outside of the environment: when the scene loads in the Player with Gravity enabled, the user falls away from the environment. In both cases, go into the Scene Editor and reposition the Entry Point.
- **Testing the Final Environment** - It's a good idea to open project in different browsers and on different versions of the operating system. Browser versions can have bugs that cause pages and environments that work fine elsewhere to have problems. This is particularly true if environment communicates with HTML elements on a web page.
- **Using World Settings** - The World Settings dialog box should be set before publish the environment. To do this, select File > World Settings (Alt+Ctrl+S). This will open the World Settings dialog box. *If wish to specify Adobe's Collaboration Server, simply enter wac://atmosphere.adobe.com in the Server URL field.*

- **Publishing an Atmosphere Environment** - Publishing completed environment is as easy as selecting File > Publish > World (Ctrl+P). This will open a file dialog box where user can save the file to the AER format. This published name can be different from the name for the project file saved in the ATMO format. The publishing process will also create an HTML file that is named the same as the AER file that will load the Atmosphere environment in a web browser. Once the file is saved, it will launch a web browser and show you the environment in the Player. If user don't want a web browser to launch when publish, user can disable this feature in the Preferences dialog box.
- **Publishing a Single Model** - In addition to publishing an entire environment, user can also publish just a single model using the File > Publish > Model menu command. Publishing a model will save it with its texture to the specified directory using the ATMO file format. It will not launch a web browser.
- **Publishing Preferences** - The Preferences dialog offers an option to Launch Web Browser on Publish. This option will cause a web browser to launch and display file when publish an Atmosphere environment—this is handy for debugging and previewing the environments. There is also an option to include a Web Page URL. This URL will be loaded in the browser when the environment is published. If the Web Page URL field is left blank, then the HTML file created by Atmosphere will be loaded in the web browser. However, if a Web Page URL is included, then the specified URL will be loaded.

CHAPTER 4
RESULT AND DISCUSSION

Generally, to begin a project, gathering information is among the important of all. At this stage, analyst should know what the system wants and requires in order meeting the clients need. Besides, the information about client itself should be considered since they are the target user that will evaluate the project later on. Both of them are important if we want to achieve the objective of the project.

Since from the beginning till now, the projects still in progress, especially in developing stage. Up till now, the main design has been completed. From time to time new features will be added to the project environments such as more collision detection, sounds and others. At the meantime, the template of the website has been finished developed and ready to be used as the final products (VR Walkthrough) finish.

4.1 Creating the VR Walkthrough

This is the main important part of the project. In this part, the project required of designing a realistic and immersive products like the actual environment. The first stage is to develop a floor plan in the Adobe Atmosphere before we began to map it. During the drawing a lot of factors shall be considered such as the wide of the area, including car porch, living room, kitchen, bedrooms and others. Below was the first stage of the development stage.

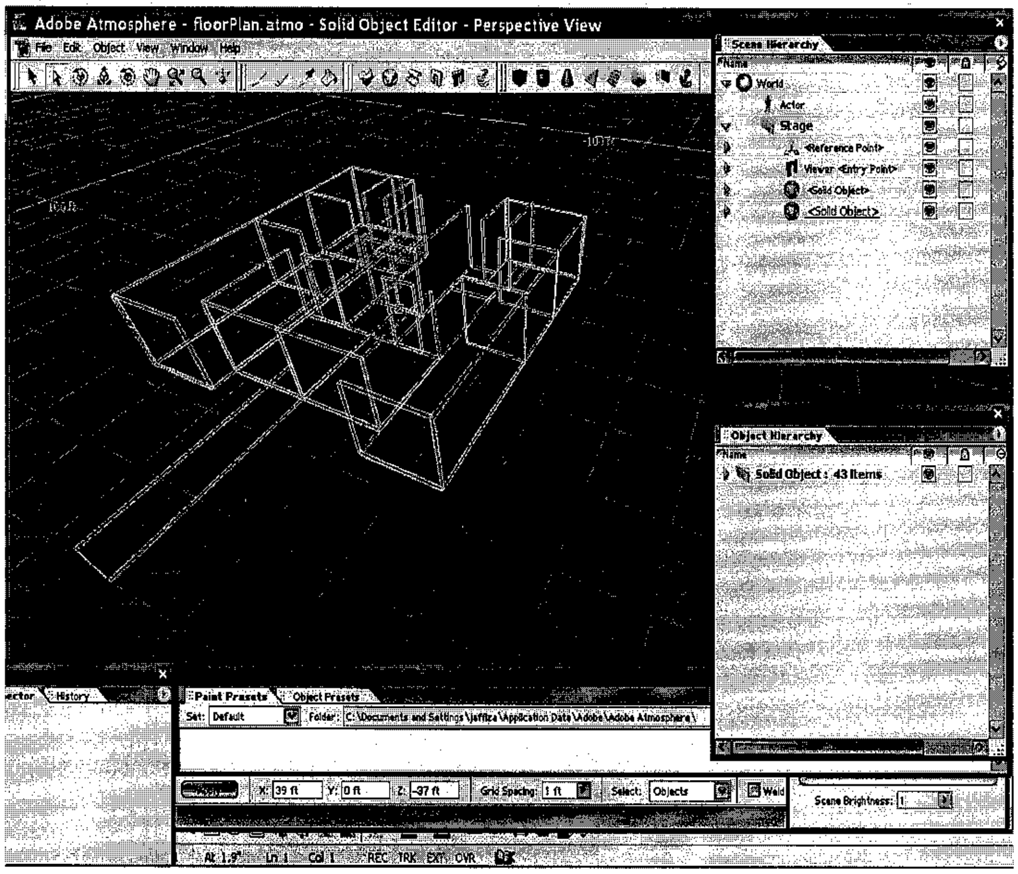


Figure 4.0: Floor Plan

Basically, Atmosphere's three editors display scene contents in either wireframe views or the Player View. The Solid Object Editor and the Scene Editor display objects using wireframe views and the Appearance Editor Displays objects in the Player View.

- Wireframe

Wireframe views show the lines that make up edges of objects and are commonly used in 3-D applications to create and modify the geometry of objects. In Atmosphere, wireframe are used to create and edit the geometry of objects and to compose scenes by applying geometric transformations to these objects. Wireframe views can be switched to eight preset positions by right-clicking in a Solid Object or Scene Editor window and selecting the Preset Position menu item. The Preset Positions are grouped in two sets in the menu: Top (F5), Front (F6), Right (F7) and Perspective (F8) in one group, and

Bottom (Shift+F5), Back (Shift+F6), Left (Shift+F7) and Isometric (Shift+F8) in the second. Note that the Shift key toggles the reverse perspective for the Top, Front and Right positions. F5 shows Top view while Shift+F5 shows bottom view etc.Example as below:

Solid Object Editor

The Solid Object Editor is used to create objects and displays them in windows that can have eight positional types. You can switch to the Scene Editor from the Solid Object Editor by toggling the Object > Edit Selected Object menu command (or you can double-click on the background of the scene).

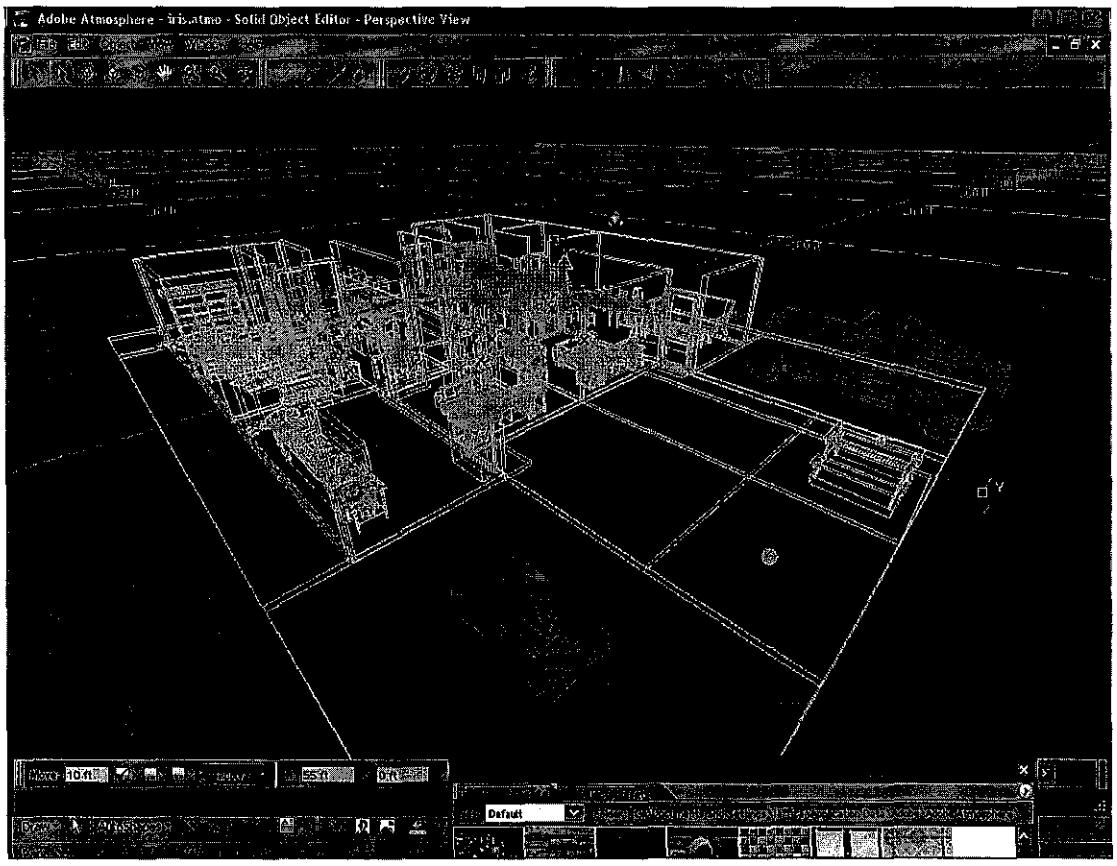

Adobe Atmosphere - iris.atmo - Solid Object Editor - Perspective View
Default

Figure 4.1: Solid Object Editor

Scene Editor

The Scene Editor is used to place scene objects and also has eight wireframe positional presets. When the Scene Editor is active, the Scene Tools toolbar is active. Notice that the background color changes when selecting Solid Object and Scene Editor. By default the Solid Object Editor's background is a dark blue and the Scene Editor's background is a dark gray. The same water fountain viewed in perspective in the Scene Editor. The Scene Editor is used to work with scene objects. You can switch between The Scene Editor and Solid Object Editor by selecting a Solid Object and choosing the Object> Edit Selected Object menu command (or by double clicking on the Solid Object in the Scene view). You can switch from Solid Object Editor to Scene Editor by double-clicking in the background away from any objects.

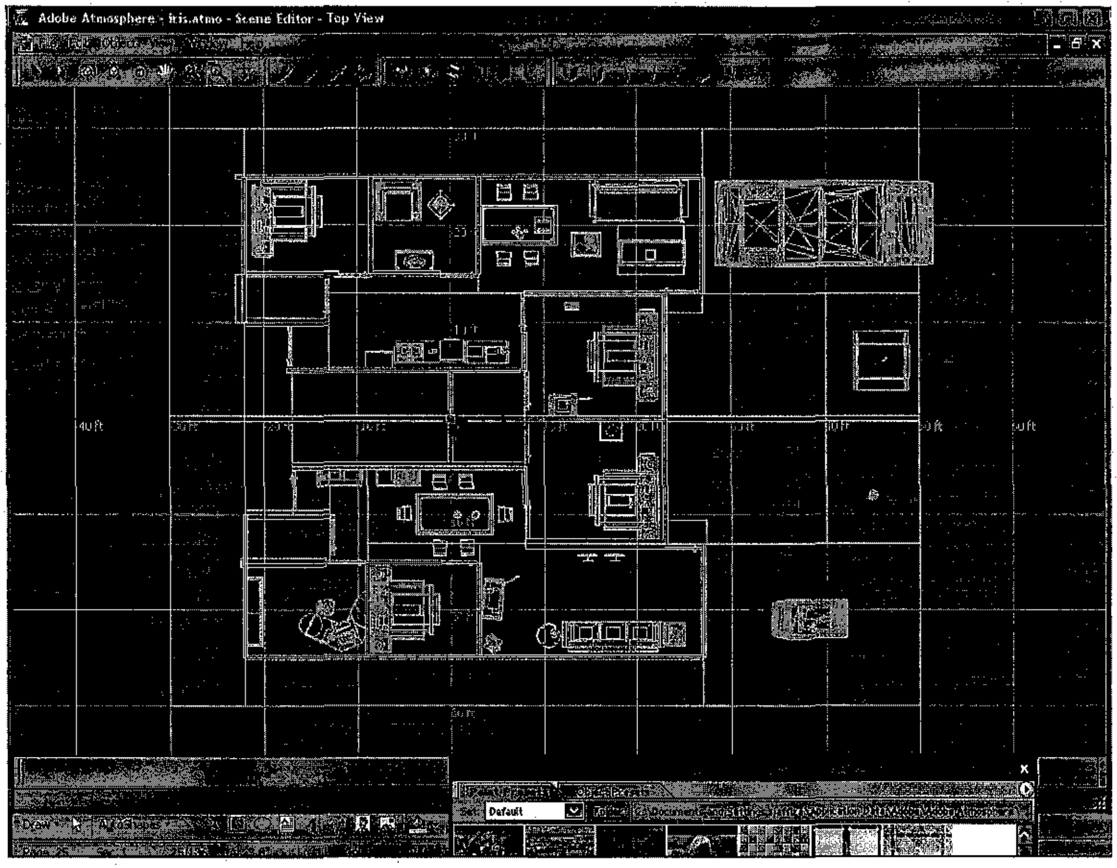


Figure 4.2: Scene Editor

- Player View

The Player View is a preview of the scene including lighting, and textures. The Player View shows the scene as it will appear in the Atmosphere Player. Scripted behaviors, run-time interaction and loaded models, sound, video, physics (with the exception of Player navigation), and dynamic lighting will not appear in the Player View. In order to test the full experience of an Atmosphere environment, the user must publish the scene and view it in a copy of the Atmosphere Player, usually within a Web browser. The Publish feature of Atmosphere will do this automatically for you. The Player View position is independent from the wireframe views and is controlled by the Navigate tool in the Tools.

Appearance Editor

The View > Player View (F4) menu command opens the Appearance Editor and displays the current scene in Player View. You can navigate around the scene just as you would in the Atmosphere Player by selecting the Navigate Tool in the Tools toolbar. The Appearance Editor also allows you to apply and remove lighting, colors and textures. *The Appearance Editor's Player View lets you see objects as they will appear in the Atmosphere Player.* Figure 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8 are taken from the VR walkthrough. All the scenes are captured from different perspectives.

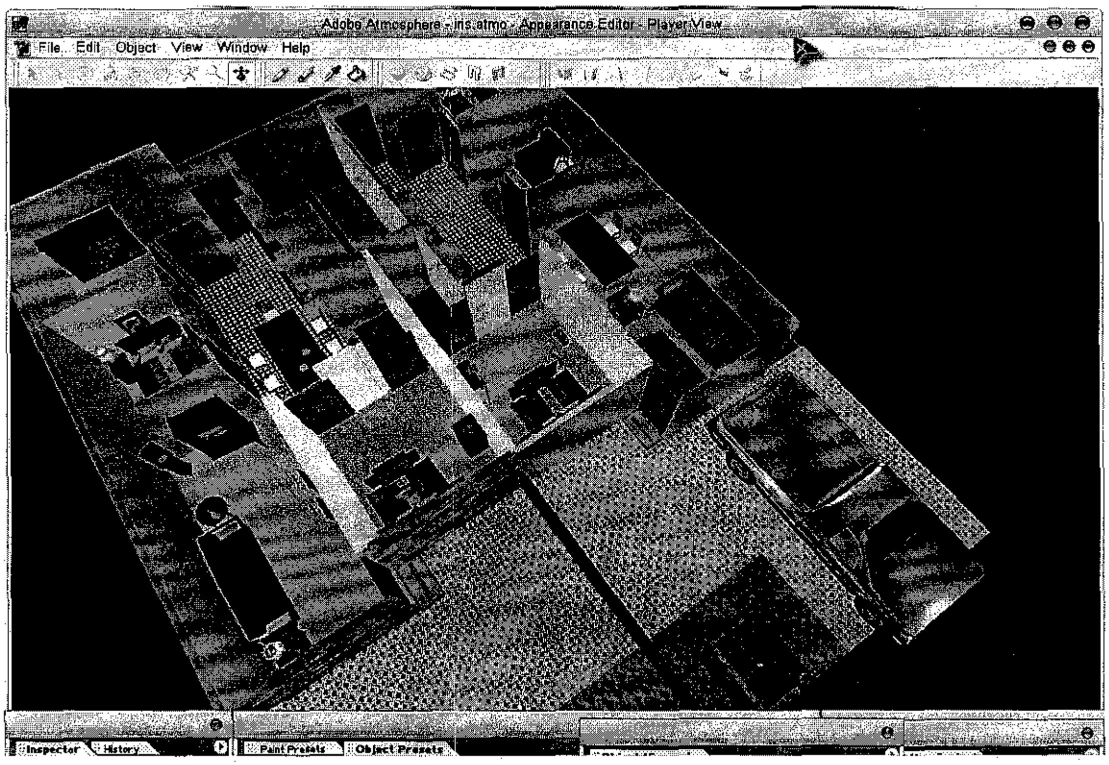

Adobe Atmosphere - iris.atmo - Appearance Editor - Player View
File Edit Object View Window Help
Inspector
History
Paint Presets
Object Presets

Figure 4.3: Upper View

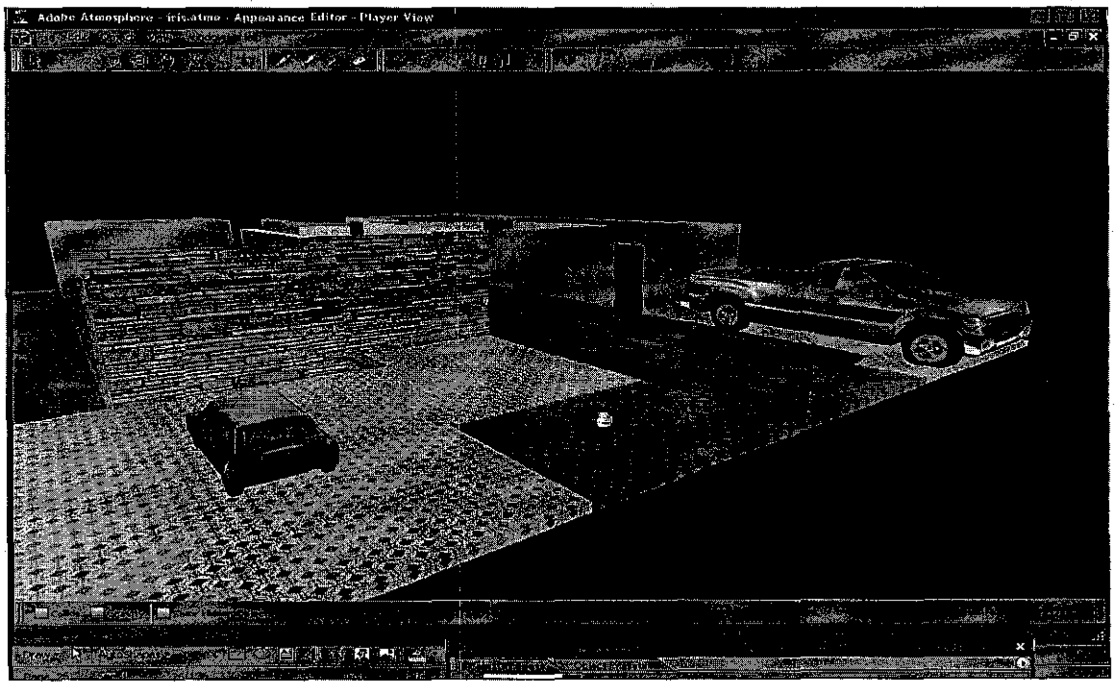

Adobe Atmosphere - iris.atmo - Appearance Editor - Player View

Figure 4.4: Front View

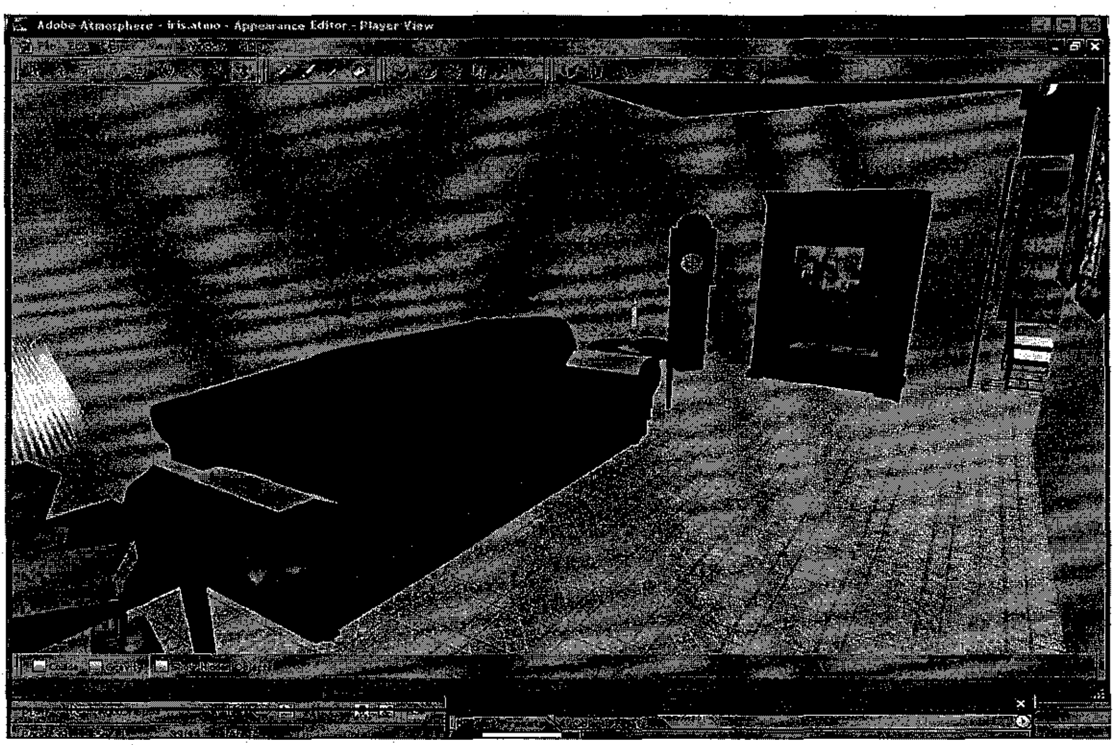

Adobe Atmosphere - iris.atmo - Appearance Editor - Player View

Figure 4.5: Living Room

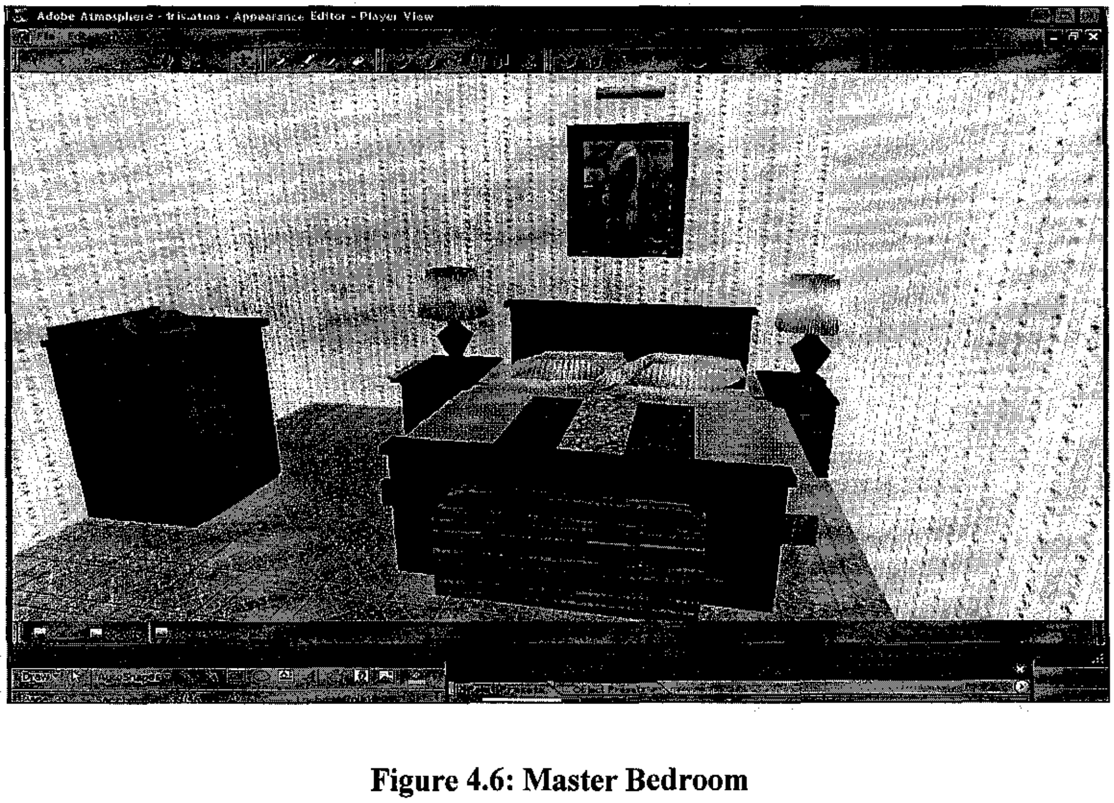

Adobe Atmosphere - iris.atmo - Appearance Editor - Player View

Figure 4.6: Master Bedroom

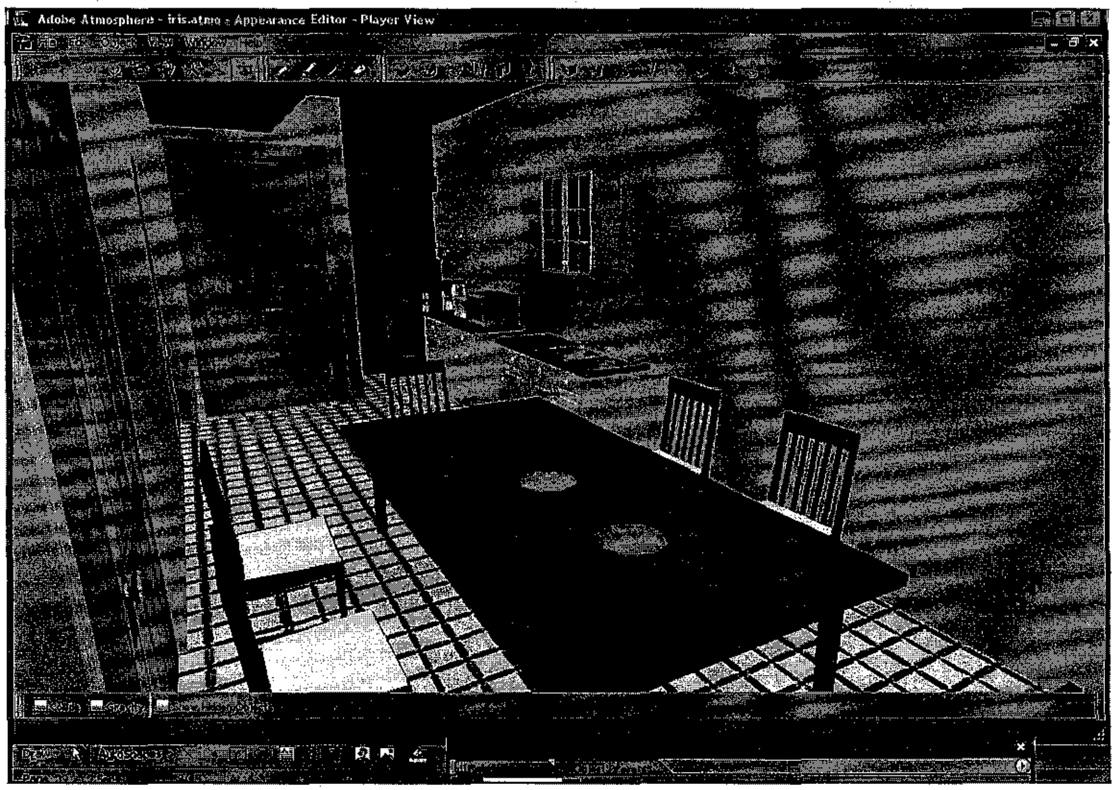

Adobe Atmosphere - iris.atmo - Appearance Editor - Player View

Figure 4.7: Kitchen

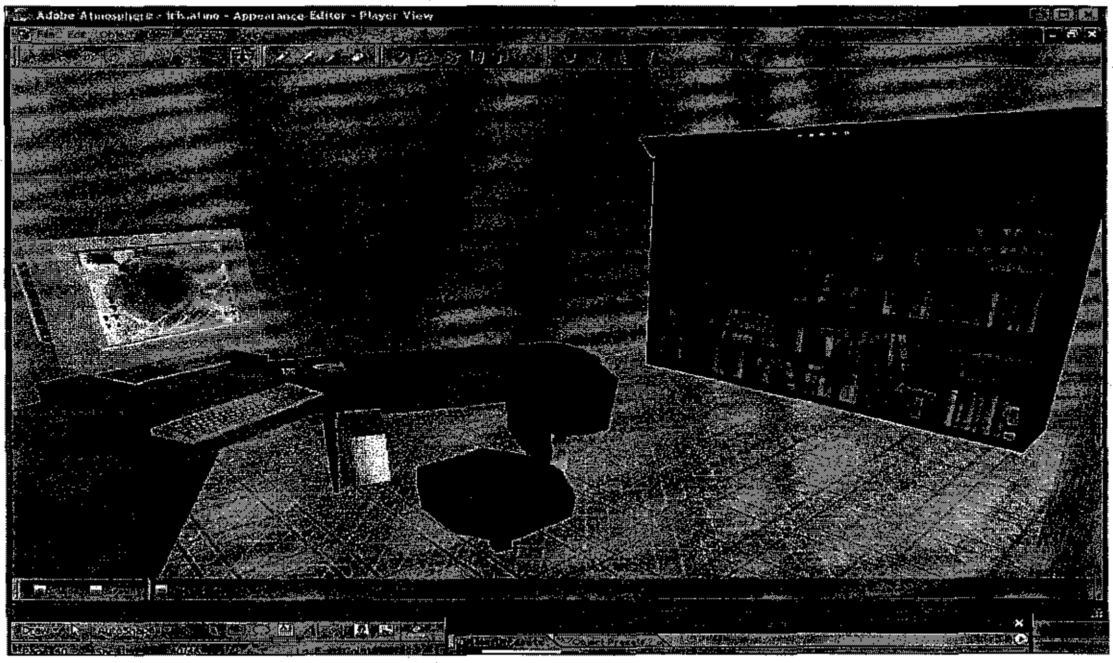

Adobe Atmosphere - iris.atmo - Appearance Editor - Player View

Figure 4.8: Other Room

At the meantime, a website has been developed in order to get ready to be used when the VR walkthrough is finished developed. The prototype will be upload to the website and this is the part of critical moment to see if the project success. The website is functioning as the medium for user that came from various places since the location is far from their current location. The website developed is a user friendly and easy to be navigated. Users can get all the information they require such as the products offered and latest residential being built through the site.

When users login to the website the first thing they will face will be the opening montage (refer to the figure 4.9) below. Basically, the montage is showing what the website is all about and its function to the society.

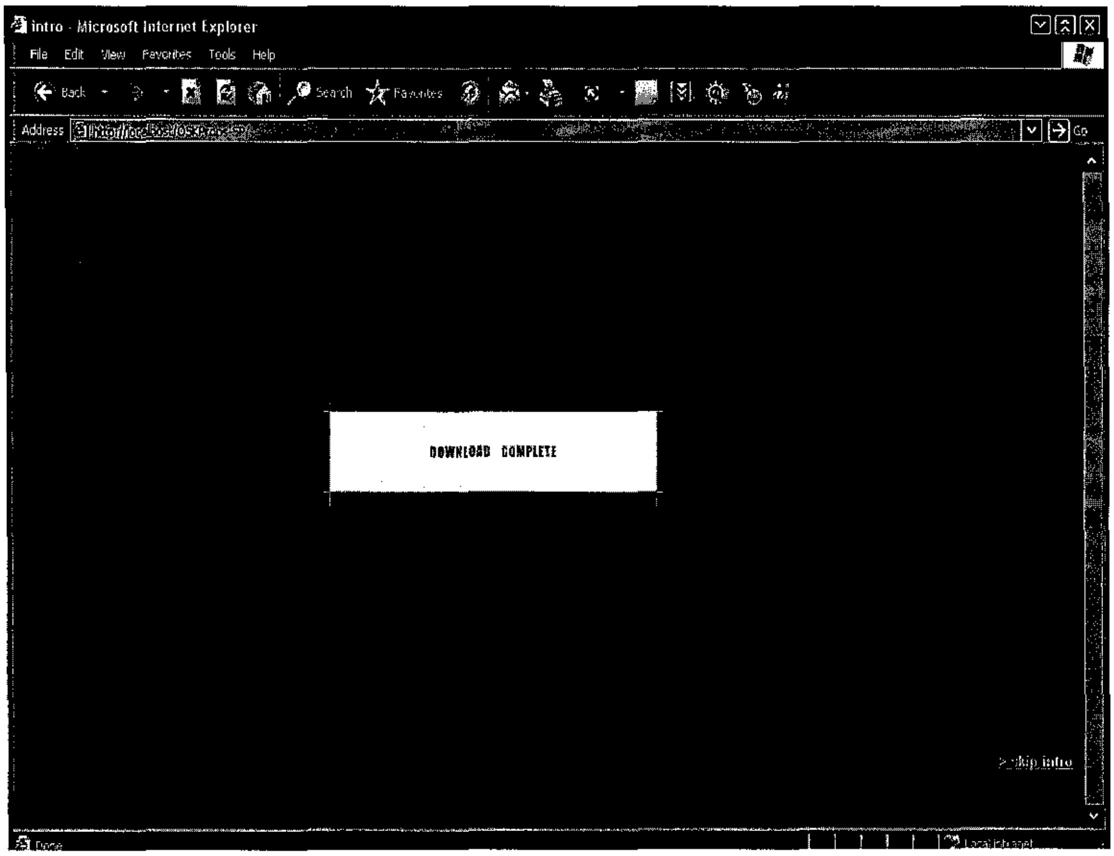


Figure 4.9: Montage

Figure 4.10 below shows the usage of the main page of the website. At this page, the top frame and left frame will be permanently located. Only, the main panel will verify depend what the term of the pages are. At the top frame, we have a toolbars of homepage, products, price, link and others. While in the left panel show about the latest news or discovery achieved by the company. For the main panel, there is a short brief about Bandar Puteri Jaya (the resident's name) and some promotion towards it. Besides, it also has a search engine for price preferences and houses base on users requirements.

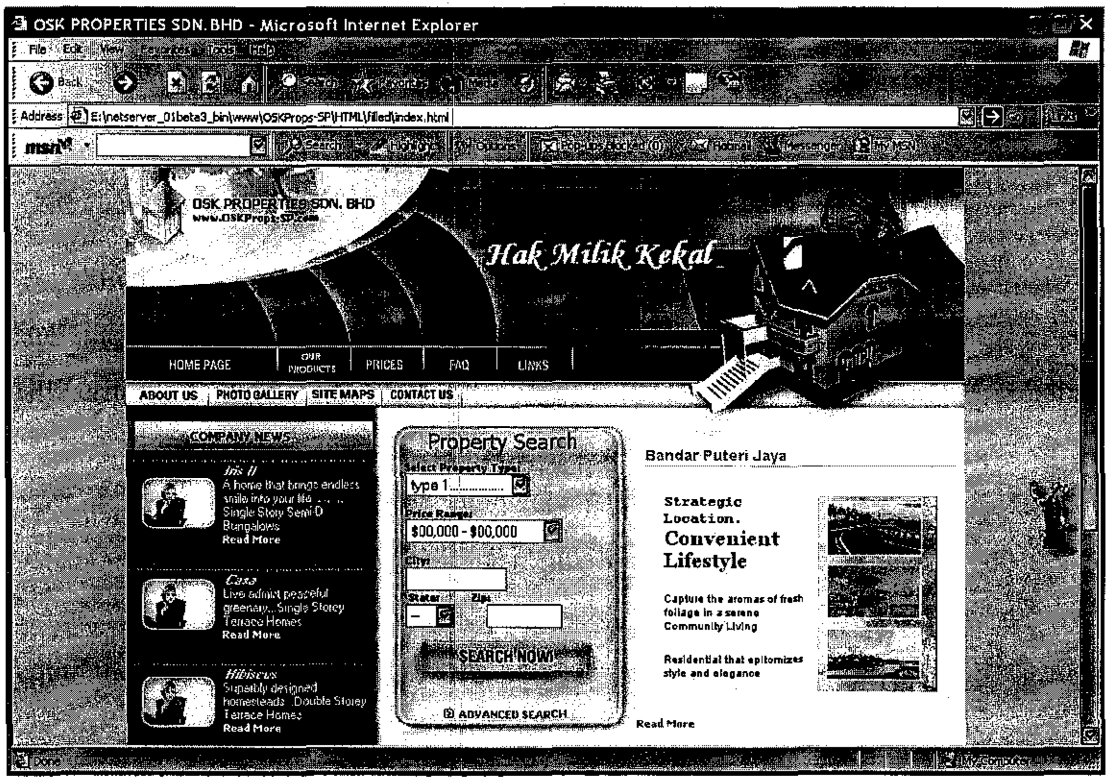
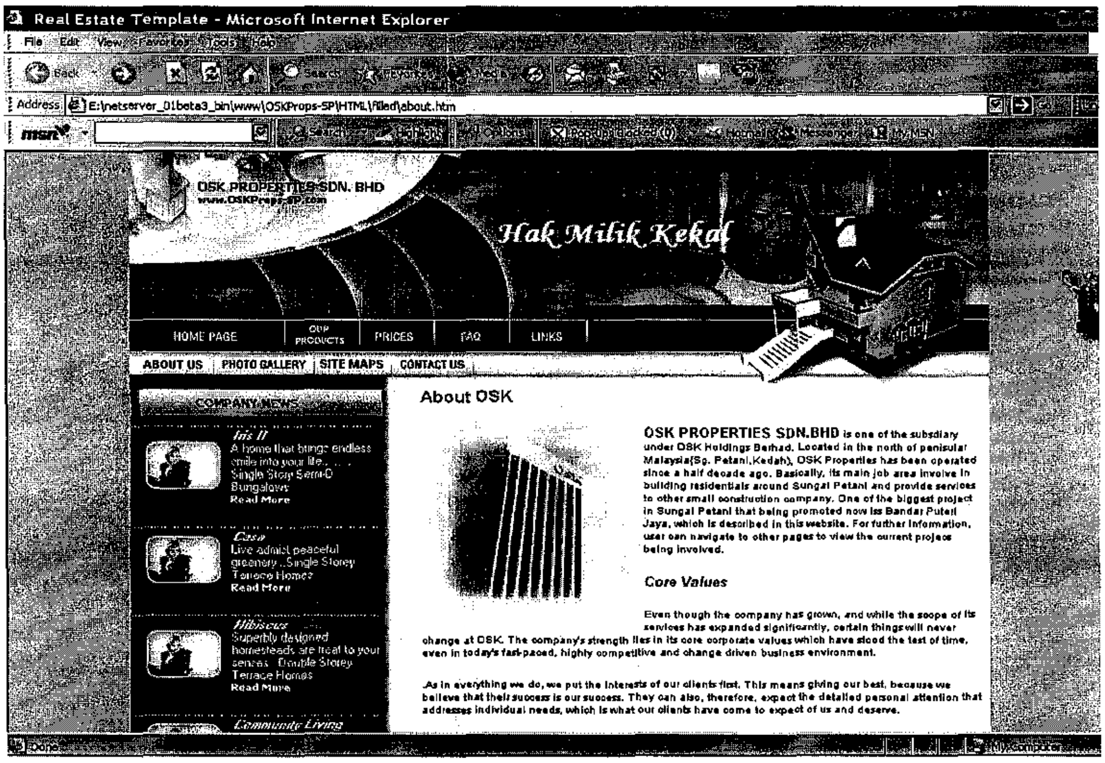


Figure 4.10: Homepage

The purpose of this page (figure 4.11) is to describe the background of the companies. Some of the features are like core values of the company, mission and vision and most important is the corporate mission. Other features like the top and left frame were the same like the main page. This is because we want to standardize all the pages for more convenient and easier to be navigating by users.


Real Estate Template - Microsoft Internet Explorer
Address E:\netserver_01beta3_bin\www\OSKProps-SP\HTML\Filled\about.htm
OSK PROPERTIES SDN. BHD
Hak Milik Kekal
HOME PAGE
OUR PRODUCTS
PRICES
FAQ
LINKS
ABOUT US
PHOTO GALLERY
SITE MAPS
CONTACT US
COMPANY NEWS
Iris II
A home that brings endless smile into your life.......
Single Story Semi-D Bungalows
Read More
Casa
Live admist peaceful greenery ..Single Storey Terrace Homes
Read More
Hibiscus
Superbly designed homesteads are treat to your senses ..Double Storey Terrace Homes
Read More
Community Living

About OSK

OSK PROPERTIES SDN.BHD is one of the subsdiary under OSK Holdings Berhad. Located in the north of penisular Malaysia(Sg. Petani,Kedah), OSK Properties has been operated since a half decade ago. Basically, its main job area involve in building residentials around Sungai Petani and provide services to other small construction company. One of the biggest project in Sungai Petani that being promoted now iss Bandar Puteri Jaya, which is described in this website. For further Information, user can navigate to other pages to view the current projecs being involved.

Core Values

Even though the company has grown, and while the scope of its services has expanded significantly, certain things will never change at OSK. The company's strength lies in its core corporate values which have stood the test of time, even in today's fast-paced, highly competitive and change driven business environment.

.As in everything we do, we put the interests of our clients first. This means giving our best, because we believe that their success is our success. They can also, therefore, expect the detailed personal attention that addresses individual needs, which is what our clients have come to expect of us and deserve.

Figure 4.11: About Us

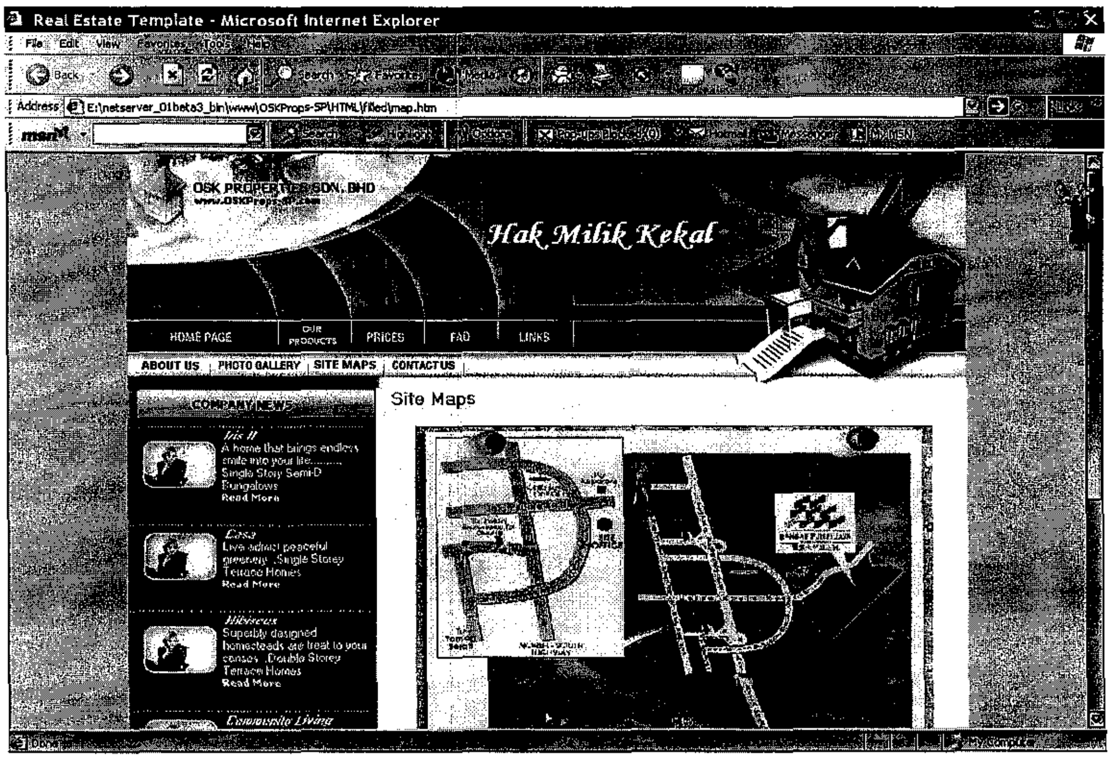
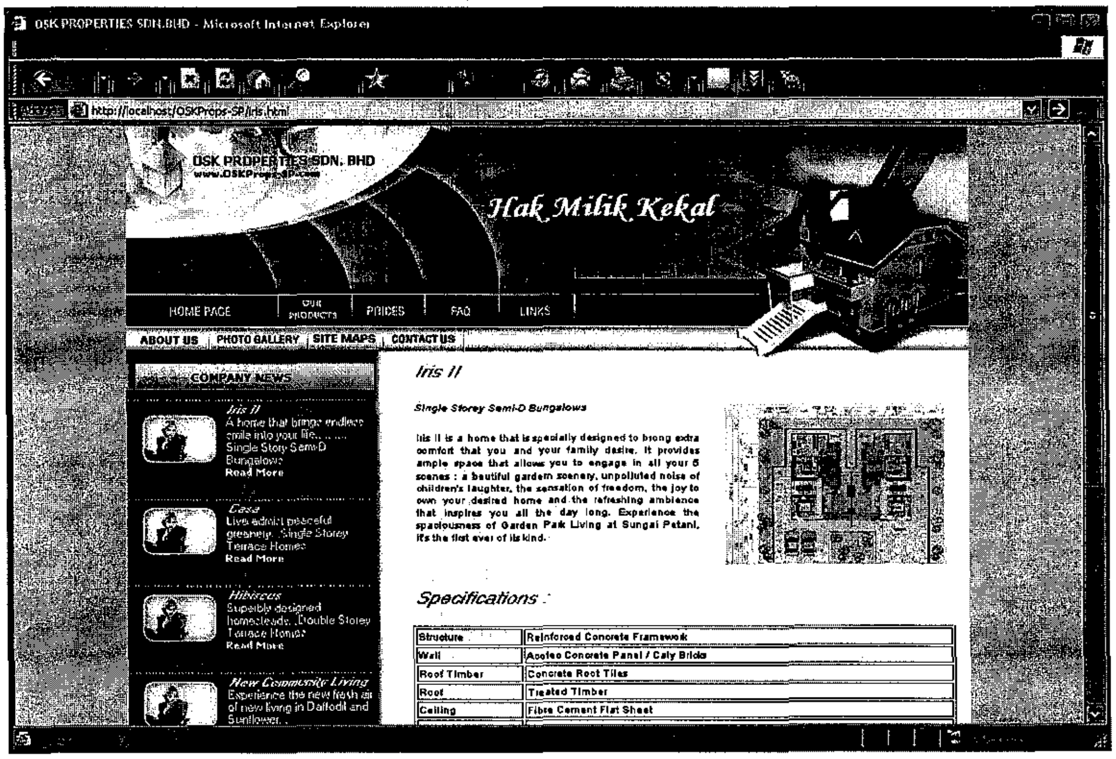


Figure 4.12: Site Maps

Above will be the final figure of the website. This is the current status of the project be done till now. Figure 4.12 shows the site map of Bandar Puteri Jaya at Sungai Petani. In order to assist the customer, the page also provides maps that covering all the access way either from Penang or the North-South Highway from the south. This page will guide the users; if there require to know the exact location of their favorite residential. Other than that, it will also provide information about the privileges and facilities around the town. This will help enough the users to imagine how Bandar Puteri Jaya looks alike.

The next figure will be the page of the product promoted. As informed in the Figure 3.3 (storyboard) before, the main point is the content, and then followed by the button for VR walkthrough at the bottom of the page. For this site, there main products have been selected, which are Iris II and Casa. Below are the examples:



Iris II
A home that brings endless smile into your life.......
Single Story Semi-D Bungalows
Read More

Casa
Live admist peaceful greenery. Single Storey Terrace Homes
Read More

Hibiscus
Superbly designed homesteads. Double Storey Terrace Homes
Read More

New Community Living
Experience the new fresh air of new living in Daffodil and Sunflower..

Iris II

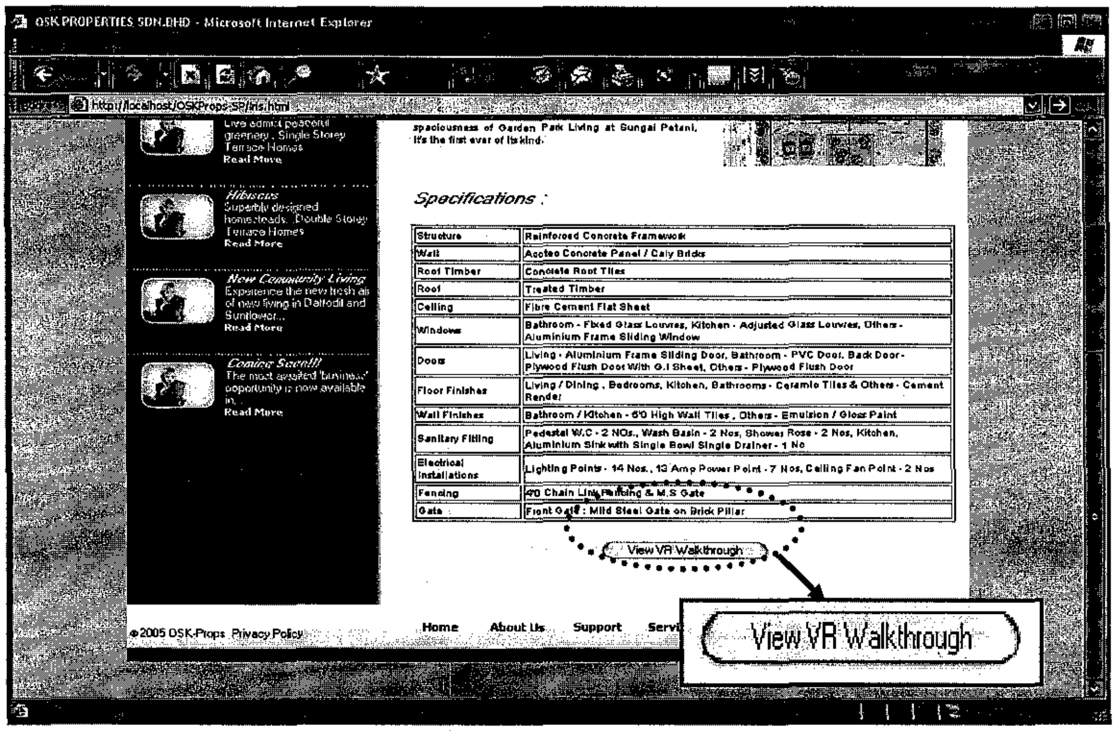
Single Storey Semi-D Bungalows

Iris II is a home that is specially designed to brong extra comfort that you and your family desire. It provides ample space that allows you to engage in all your 5 scenes : a bautiful gardem scenery, unpolluted noise of children's laughter, the sensation of freedom, the joy to own your desired home and the refreshing ambience that inspires you all the day long. Experience the spaciousness of Garden Park Living at Sungai Petani, it's the first ever of its kind.

Specifications :

Structure	Reinforced Concrete Framework
Wall	Acotec Concrete Panel / Caly Bricks
Roof Timber	Concrete Roof Tiles
Roof	Treated Timber
Ceiling	Fibre Cement Flat Sheet

Figure 4.13: Iris (part 1)



OSK PROPERTIES SDN.BHD - Microsoft Internet Explorer

http://localhost/OSKProps-SP/iris.html

Live admist peaceful greenery , Single Storey Terrace Homes
Read More

Hibiscus
Superbly designed homesteads , Double Storey Terrace Homes
Read More

New Community Living
Experience the new fresh air of new living in Daffodil and Sunflower...
Read More

Coming Soon!!!
The most awaited 'business' opportunity is now available in...
Read More

spaciousness of Garden Park Living at Sungai Petani, It's the first ever of its kind.

Specifications :

Structure	Reinforced Concrete Framework
Wall	Acotec Concrete Panel / Caly Bricks
Roof Timber	Concrete Roof Tiles
Roof	Treated Timber
Ceiling	Fibre Cement Flat Sheet
Windows	Bathroom - Fixed Glass Louvres, Kitchen - Adjusted Glass Louvres, Others - Aluminium Frame Sliding Window
Doors	Living - Aluminium Frame Sliding Door, Bathroom - PVC Door, Back Door - Plywood Flush Door With G.I Sheet, Others - Plywood Flush Door
Floor Finishes	Living / Dining , Bedrooms, Kitchen, Bathrooms - Ceramic Tiles & Others - Cement Render
Wall Finishes	Bathroom / Kitchen - 6'0 High Wall Tiles , Others - Emulsion / Gloss Paint
Sanitary Fitting	Pedestal W.C - 2 NOs., Wash Basin - 2 Nos, Shower Rose - 2 Nos, Kitchen, Aluminium Sink with Single Bowl Single Drainer - 1 No
Electrical Installations	Lighting Points - 14 Nos., 13 Amp Power Point - 7 Nos, Ceiling Fan Point - 2 Nos
Fencing	4'0 Chain Link Fencing & M.S Gate
Gate	Front Gate : Mild Steel Gate on Brick Pillar

Figure 4.14: Iris (part 1I)

Basically, this project also explain the important having a manageable content management in maintaining the virtual 3D web world. For this project, a prototype of a virtual 3D web using Adobe Atmosphere platforms will be developed. This is to show how having a manageable content can help the web administrator in maintaining the virtual 3d world especially when editing the world object is required.

4.2 Data Analysis

After implementing the data collection methods, finally it has produced a result for this research. Basically, the objectives of the data analysis are

- To stress the importance of getting a feel for goodness of data that we have been collecting, then
- To understand and apply the different types of analyses and tests on the variables and lastly
- To interpret the results

From the starting point; begun implementing the data collection methods since two weeks ago by distributing the research questions form among students in UTP. Rather than that, an experimental design has been done during the classes in order to support the hypothesis regarding this research. After obtaining all the data required, finally it comes out with a graphics representations that based on the research conducted.

From the Figure 4.14 below, it is proof that the major percentage 33% of the students said that they agree with the entire questions given (refer to the appendix to view the questions). It means that from the survey conducted, it proved that most of the students are very comfortable with walkthrough on online. Followed by is 27% which represents of strongly agreeing. As the second preferences, it has strength the result in the agreeing the products offer by the developer. The third preferences are 20% which is neither agree or disagree with the questions asked. In this section, the students is rather confuse or not sure what they wanted in the products, which means some part of the application is not meet their requirement or satisfaction.

The fourth preferences is about 13% which represent the disagree decision from the student. The result of this surveys is actually based on how much question that been selected by disagree decision. From the percentage, it shows that only several students are not agreeing with the questions asked in the survey. The poorest preferred decision in the survey is only 7% percent, which means this is the least decision made by the students. From the percentage, it proved that only a one or two questions are not strongly disagree by the students. For clearer visualization, see the graphics representations of the survey conducted below.

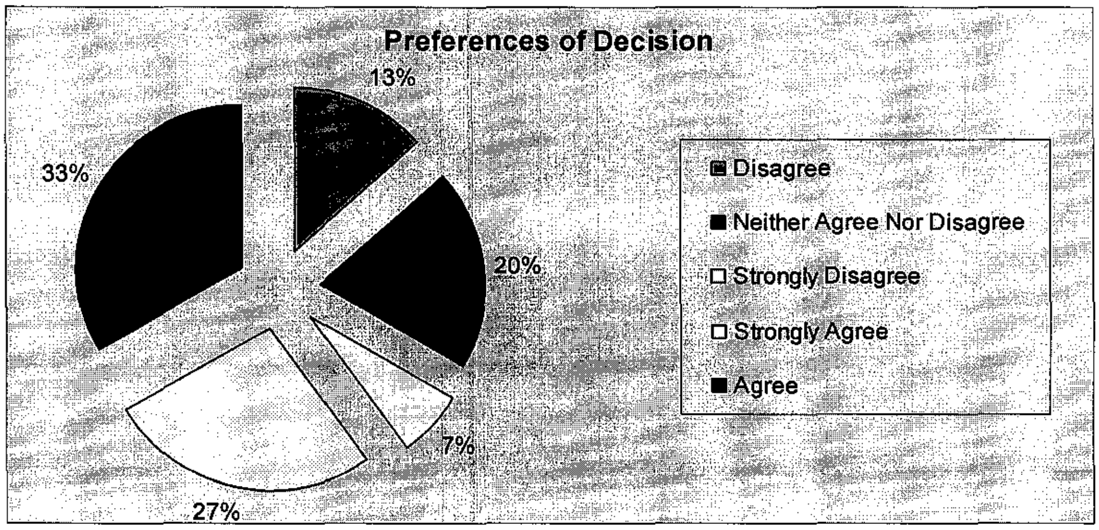


Figure 4.15: Preferences of Decision

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

By incorporating VR walkthrough on the website, the houses could be presented in a more elaborative way. With this, the developer is capable to introduce to the public about the initial conceptual design through detailed design, planning and preparation, until the construction completion. In this project, an immersive virtual environment will be developing where it is interactive and considerably realistic. The VE is then embedded into web pages whereby the customers can view it. At the end of the project, I'll hope that the product will meet the mission of an exploration into the role of multimedia, content and the customers who participates in the buying process.

5.2 Recommendation

For the recommendation part, a lot of enhancement should be done to the project. Planning of the project is very important especially when conducting a short-time project like this. As for me, the first recommendation will be time taken for developing a project should be longer. Meaning, only this semester the final year project is conducted in one (1) semester, others were in two (2) semesters. Since there will be much time for next year student, it's recommended that this kind of project should be added with a lot more objects and interactive features. This is because, the immersiveness of the environment will depend on how real the feeling was, and else if the project has all the support collusion detection and sounds in the environment, then it will be much advantage.

In the other hand, it's recommend to plan the work in detail before attempted to conduct any development. For this project, the respective properties developers have been contacted in order to enter the site location and obtain all the information needed. At this time, it's better to discuss what they need or required in the project. All this kind information will help a lot in the developing stage. Work will be insufficient if there isn't much information about the products they're promoting at. Nowadays, a lot of software has been used in designing interactive multimedia development. As for recommendation, a research need to be done first in order to find the appropriate software before begin the development. Search on several potential software, then compare the pros and cons to each other. To assist more, just ask the previous student or lecturers who expert in this field.

Lastly, if the project still incomplete, the result of the project must be shown even a portion of it. For this project, it's recommended to show the final stage of integration between the published products with web browser. Although the product is not fully furnished with script or others, the proof of integration between web-based and virtual reality (VR) will be enough to convince the internal or external examiner.

REFERENCES

Mahoney, D.P.1994, Walking through Architectural Designs, *Computer Graphics World*, v17 n6 p22 (7)

Gibson, G. 1996, *Perceptual and Visualization*, Information Science and System, Morgan State University, USA

Wicken, J. and Baker, N. 1994, *VR Component in Reality Aspects*, Department of Chemistry, Imperial College, London

Henriette S. M. Cramer, Evers, V., Zudilova, E.V. and Sloot, P.M.A. 2004, *Context Analysis To Support Development Of Virtual Reality Applications*, Springer-Verlag London Limited 2004

Hartley, D. and Churcher, N. 2004, *Virtual Worlds for Web Site Visualization*, Software Visualization Group, Department of Computer Science, University of Canterbury, Private Bag 4800, Christchurch, New Zealand

Persiani, F. 2001, *Semi-Immersive Virtual Reality*, DIEM, University of Bologna, Italy

Roseendaal, T. 2004, Blender, <http:// www.blender3d.org>

Mitchell, K. 1996, Virtual Reality, <http://www.VirtualReality.com>

Deutch, D. 2001, Universality, <http:// www.virtualreality@Everything2_com.html>

APPENDIXES

Appendix 1: QUESTIONAIRES

Using the scale below, state the extent to which you agree with each of the following statements in measuring the efficiency of navigating the VR walkthrough on the website.

Strongly Disagree	Disagree	Neither agree Nor disagree	Agree	Strongly agree
1	2	3	4	5

1. Save time and costs rather than visiting the show houses at the site.	1	2	3	4	5
2. Place a booking on the website is more faster than the usual way of buying houses.	1	2	3	4	5
3. Increase understanding about the house design when navigating VR walkthrough.	1	2	3	4	5
4. The feeling of immersiveness is likely close to the real environment.	1	2	3	4	5
5. Feel more interesting and joyable when navigate the walkthrough.	1	2	3	4	5
6. It's more satisfy to navigate the walkthrough rather than looking still picture or visit the promotion centre.	1	2	3	4	5

7. VR walkthrough is an interesting marketing strategy in promoting product.	1	2	3	4	5
8. Website marketing is potential in term of global and can insert many informative data as we can.	1	2	3	4	5
9. User gain more control in term of selecting the house that they're interested before decide to buy.	1	2	3	4	5
10. By using online marketing, the construction company can update their activities regularly.	1	2	3	4	5

Appendix 2: KEYBOARD SHORTCUTS

Learning and using the keyboard shortcuts will help you become much more efficient with Atmosphere. The following list includes all the keyboard shortcuts for Adobe Atmosphere.

File > New Ctrl+N
File > Open Ctrl+O
File > Close Ctrl+W
File > Save Ctrl+S
File > Save As Shift+Ctrl+S
File > World Settings Alt+Ctrl+S
File > Publish > World Ctrl+P
File > Exit Ctrl+Q
Edit > Undo Ctrl+Z
Edit > Redo Shift+Ctrl+Z
Edit > Duplicate Ctrl+D
Edit > Clear Delete
Edit > Clear Transform Shift+Ctrl+T
Edit > Extrude Walls From Floor Shift+Ctrl+E
Edit > Deselect All Shift+Ctrl+A
Edit > Select All Ctrl+A
Edit > Preferences Ctrl+K
Object > Group Ctrl+G
Object > Ungroup Shift+Ctrl+G
Object > Hide > Selection Ctrl+H
Object > Hide > All Alt+Ctrl+H
Object > Show All Shift+Ctrl+H
Object > Lock > Selection Ctrl+L
Object > Lock > All Alt+Ctrl+L
Object > Unlock All Shift+Ctrl+L
Object > Edit Script Ctrl+E
View > New View Shift+Ctrl+N
View > Preview F4
View > Top F5
View > Front F6
View > Right F7
View > Perspective F8
Access Window menu Alt+W
Access Help menu Alt+H
Move to next menu Right arrow
Move to previous menu Left arrow
Move up menu list Up arrow
Move down menu list Down arrow
Execute selected menu command Enter
Move focus to next palette control Tab
Move focus to previous palette control Shift+Tab
Simulate click on button with focus Enter
Move forward Up arrow
Move backward Down arrow
Turn to right Right arrow
Turn to left Left arrow
Move upward Shift+up arrow
Move downward (with Gravity disabled) Shift+down arrow
Tilt view upward Ctrl+up arrow
Tilt view downward Ctrl+down arrow
Strafe right Shift+right arrow
Strafe left Shift+left arrow
View > Bottom Shift+F5
View > Back Shift+F6
View > Left Shift+F7
View > Isometric Shift+F8
View > Zoom In Ctrl++
View > Zoom Out Ctrl+-
View > Fit All Ctrl+0
View > Fit Selected Shift+Ctrl+0
View > Grid Ctrl+'
View > Snap to Grid Shift+Ctrl+'
View > Guidelines Ctrl+;
View > Shadows Shift+Ctrl+;
View > Reset Views Shift+Ctrl+R
Help > Atmosphere Help F1
Help > Javascript API Documentation F2
Help > Lighting Overview F3
Selection Tool V
Direct Selection Tool A
Rotate Tool R
Scale Tool S
Orbit Camera Tool C
Hand Tool H or spacebar
Dynamic Zoom Tool D
Zoom Tool Z
Navigate Tool N
Remove Texture Tool E
Sample Texture Tool I
Apply Texture Tool K
Access File menu Alt+F
Access Edit menu Alt+E
Access Object menu Alt+O
Access View menu Alt+V

Appendix 3: EXAMPLES OF JAVASCRIPT API

Utility Functions

animators

```
chat.print ("There are (at most) " + animators.size + " animators.");
```

context

indicates the context in which the script is being run:

worldContext The script is attached to a world that has been opened

localAvatarContext The script is attached to your avatar, running locally

remoteAvatarContext The script attached to your avatar, running on a remote client

```
if (context == localAvatarContext)
{
chat.print("This is a LOCAL avatar script.");
}
else if (context == RemoteAvatarContext)
{
chat.print("This is a REMOTE avatar script.");
}
else
{
chat.print("This is not an avatar script.");
}
```

localAvatarContext

the value of the 'context' global in an Avatar script which is running on the client that owns the Avatar.

remoteAvatarContext

the value of the 'context' global in an Avatar script which is running on another user's client.

worldContext

the value of the 'context' global in a world script.

version

Returns the version number of the application.

```
// Some features didn't exist before then...
if (version < 1.2) { ... }
```

Global Methods

addAnimator(object)

```
foo = new Object;
foo.timestep = function(t, deltaT)
{
chat.print("This timestep is at " + t + " seconds.");
chat.print("Last timestep was at " + (t - deltaT) + " seconds.");
}
addAnimator(foo);
```

removeAnimator(object)

Removes the given object from the list of animators. If the object is in the animators list more than once, onlyone instance of it is removed.

```
removeAnimator(foo);
```

dump(object)

Displays the properties of the object in the chat output window.

```
dump(fog);
```

dir(object)

Displays a hierarchical list of the object's children in the chat output window.

```
obj = theStage.getSolidObject(0);
prim = obj.rootPrimitive;
dir(prim);
```

path(object)

Returns the absolute path of an object based on parent links.

```
prim = theStage.getPrimitive("somePrimitive");
```

```
primPath = path(prim);
chat.print(primPath);
```

PluginCommand(stringCommand, 0, 0)

```
//automatically hide the Atmosphere Chat pane
document.all.MetaCtl0.PluginCommand("eval(application.chatPaneVisible = false);", 0, 0);
//move a world object using a variable value from the webpage
document.all.MetaCtl0.PluginCommand("eval(box.position = Vector(box.position[0], " +
pageValue + ", box.position[2] ) );", 0, 0);
```

sendJS(stringCommand)

```
// Open the URL in a browser window
sendJS("window.open ('http://www.adobe.com', '', '');");
// Pop up a JavaScript alert window
sendJS("alert(\"Hello World!\");");
// Update a webpage form checkbox with the Atmosphere current state
sendJS("document.forms['myForm'].chatPaneVis.checked = " + application.chatPaneVisible);
```

$.gc()

```
//delete and recreate a new array of box SceneGroups
function deleteBoxes()
{
//delete the fi rst group of boxes
for ( i=0; i<50; i++ )
{
delete boxes[i].SceneGroup;
}
//force memory clean up
$.gc();
//recreate a new box array
for ( i=0; i<50; i++ )
{
boxes[i].SceneGroup = SceneGroup("./box.aer");
}
}
```

Callbacks

timestep(t, deltaT)

```
box = SceneGroup("./box.aer").add();
box.timestep = function(t, deltaT)
{
box.orientation = Rotation('y', t);
}
addAnimator(box);
```

onLoad()

```
box = SceneGroup("./box.aer").add();
box.onLoad = function(now)
{
chat.print("onLoad called at: " + now);
}
addAnimator(box);
```

application

```
// Print out properties associated with application
dump(application);
```

type

```
// prints "Application" to the chat console
chat.print(application.type);
```

controlKeyDown

A boolean fl ag that is true when the control key is down.

if (application.controlKeyDown) { ... }
shiftKeyDown
A boolean fl ag that is true when the shift key is down.
if (application.shiftKeyDown) { ... }
leftArrowKeyDown
A boolean fl ag that is true when the left-arrow key is down.
if (application.leftArrowKeyDown) { ... }
rightArrowKeyDown
A boolean fl ag that is true when the right-arrow key is down.
if (application.rightArrowKeyDown) { ... }
upArrowKeyDown
A boolean fl ag that is true when the up-arrow key is down.
if (application.upArrowKeyDown) { ... }
downArrowKeyDown
A boolean fl ag that is true when the down-arrow key is down.
if (application.downArrowKeyDown) { ... }
mouseButtonDown
A boolean fl ag that is true when the left mouse button is down.
if (application.mouseButtonDown) { ... }
chatPaneVisible
Controls display of the chat pane in the Player window area.
application.chatPaneVisible = false; //turn it off for maximum viewable area
splitPercentage
Specifi es the percentage of the Player window area which should be occupied by the chat pane.
application.splitPercentage = 20; // set to 20 percent

toolbarVisible
Controls the display of the toolbar which appears at the lower edge of the Player area.
application.toolbarVisible = false; //turn it off for maximum viewable area
Methods
getView(index)
Returns the View object selected by the value of index. Currently the only valid value for index is 0.
view = application.getView(0);
dump(view);
getViewCount()
Returns the number of views in the application. Currently will always return 1.
numViews = application.getViewCount();
chat.print("There is/are " + numViews + " view(s).");
View
The View module is an object that represents a 2-D rendered window that is being used to image a 3-D world. The module properties include the size of the rendered area and the current mouse coordinates relative to the view. The View module also contains an array of cameras that are being used to image the world onto the 2-D view plane
type
The type of the application returned at "View".
if (foo.type == 'View') { ... }
imageWidth
The width of the current rendered view in pixels.
view = application.getView(0);
chat.print("Current rendered image width = " + view.imageWidth);
imageHeight
The height of the current rendered view in pixels.
view = application.getView(0);
chat.print("Current rendered image height = " + view.imageHeight);
mouseX
screen, with the value increasing as the cursor goes to the right.

view = application.getView(0);
stageModel.getSolidObject(0).rootPrimitive.onClick = function()
{
chat.print("You clicked down at X-coordinate: " + view.mouseX);
}
mouseY
The Y coordinate of the pixel that the mouse cursor is over in the rendered view. 0 represents the top of the
screen, with the value increasing as the cursor goes down the screen.
view = application.getView(0);
stageModel.getSolidObject(0).rootPrimitive.onClick = function()
{
chat.print("You clicked down at Y-coordinate: " + view.mouseY);
}
mouseInitialized
This is a boolean property which will return 'true' when the mouse is over the render view area.
view = application.getView(0);
if (view.mouseInitialized) { /* ready to track mouse */ }
Methods
getCamera(index)
Function that returns a reference to the camera for given numerical index
camera = view.getCamera(0);
getCameraCount()
Function that returns the number of cameras associated with rendering this view. (Currently exactly one.)
numCameras = view.getCameraCount();
triggerRedisplay()
Forces the view to be fully redrawn during the next time step. This may be necessary when changing the appearance of Viewpoint objects.
view = application.getView(0);
view.triggerRedisplay();
Button
Button(label)
creates a Button with the specifi ed label (optional). The Button will not be added to the Control Panel until you
call add().
takePicture = Button("Say Cheese");
Toggle(label)
creates a Toggle with the specifi ed label (optional). The Toggle will not be added to the Control Panel until you
call add().
fogToggle = Toggle("Fog Enabled");
Properties
state
true if the button or toggle is down/checked, false if it is up/unchecked.
// Init the toggle to refl ect the current fog state.
fogToggle.state = fog.active;
enabled
If false, the button or toggle will not respond to clicks.
// Don't let the user change the fog right now.
fogToggle.enabled = false;
label
the button or toggle's current label
button.label = "Try Again!";
Methods
add()

adds the button or toggle to the control panel associated with the current world. The toggle itself is returned as a
convenience.
fogToggle = Toggle("Fog Enabled").add();
remove()
removes the button or toggle from whatever control panel it is currently in.
fogToggle.remove();
Callbacks
onClick(state)
is called when the button is pressed or when the toggle state changes (either due to the user clicking on it, or if toggle.state is changed by JavaScript code).
```
fogToggle.onClick = function(state)
{
fog.active = state;
}
```
Callbacks
onChange(value)
Is called when the slider's value changes (either because the user moves it or if slider.value is changed by the
script).
fogNearSlider.onChange = function(value) { fog.near = value; }
PrintDevice & Object Synchronization
The PrintDevice is an object which is used to direct text strings to a destination, such as the chat window.
Global Properties
chat
A PrintDevice representing the chat window in the Atmosphere Player.
chat.print("Hello World");
Properties
avatarID
A GUID for your avatar in the current world. This property will allow you to send messages to your remote avatar more easily. Note that it is not possible to obtain the GUID of another avatar.
myCurrentAvatarID = avatarID;
isConnected
This property will return true once a connection with the server has been established.
if (isConnected) { ... }
worldID
A GUID for the world you are currently visiting.
myCurrentWorldID = worldID;
Methods
input(a, b, c, ...)
print(a, b, c, ...)
Prints the specifi ed strings or values. Note that this method is not available for your Remote Avatar script.
status.print("All is well.");
createSharedObject(str objectName)

```
chat.onConnect = function()
{
chat.createSharedObject("myGlobe");
}
```
getSharedProperty(str objectName, str propertyName)
This method is a local check of the property value; it does not communicate with the server
chat.getSharedProperty("myGlobe","isSpinning");
requestSharedProperty(str objectName, str propertyName)

setSharedProperty(str objectName, str propertyName, str Value)

Use this method to set the shared property value. This method must be called only after ‘objectName’ and ‘propertyName’ are known to exist.
chat.setSharedProperty(“myGlobe”,”isSpinning”, “true”);
Callbacks
filterInput(textInput)

```
{
if (msg.match(/^PING!/))
{
chat.print('PONG!');
return true; //Don't send the "PING!" on to the server.
}
else
{
return false; //Otherwise, let the message pass.
}
}
```

filterOutput(textOutput)

```
chat.fi lterOutput = function(msg)
{
// If the message string contains "MyCoolWave", trigger my
// remote avatar to wave, and strip the "MyCoolWave" text
// out of the message before sending it to the chat
if (msg.match(/MyCoolWave/))
{
doWave(); // Run my wave
msg = msg.replace(/MyCoolWave/, "");
// Strip out "MyCoolWave"
}
return msg;
}
```

onConnect( )
A callback which is processed when server connection is established. Call createSharedObject() in this function
to establish the objects that will be synchronized. Note that timing limitations exit for script processing, and
attempting to induce extensive server callbacks may fail.

```
chat.onConnect = function()
{
chat.createSharedObject("myGlobe");
}
```

onSharedPropertyChange(str objectName, str propertyName, str Value)
This is the main callback which occurs when any visitor updates a property value for any shared object.
chat.onSharedPropertyChange = function(objName, objProp, objValue) { ... }
Math
Rotation
Global Methods
Rotation(various...)
Creates a new Rotation(rotational transformation object). The following parameter styles are understood:
• Rotation() - The identity Rotation(no rotation at all)
• Rotation(‘X’, .7) - Rotate by .7 radians around X.
• Rotation(‘Y’, .3) - Rotate by .3 radians around Y.
• Rotation(‘Z’, .9) - Rotate by .9 radians around Z.
• Rotation(‘XZY’, .7, .9, .3) - Rotate around Y by .3, Z by .9, then X by .7
(Rotations are processed from right to left.)
• Rotation(vec, angle) - Rotate around Vector ‘vec’ by ‘angle’ radians.
• Rotation(r1, r2) - Rotate by Rotation r2, then by Rotation r1 (think ‘r1 of r2’)

• Rotation('X', .7, r2) - Rotate by Rotation r2, then around X by .7
• Rotation(r1.toString()) - decoding from an encoded string.
• Rotation(r1, r2.toString(), 'X', .7, ...) - Any number of concatenations are allowed.
critter.orientation = Rotation('Y', critter.yRotation);
Properties
inverse
The inverse (opposite) of the rotation.
// Relative rotation from a to b
rel = Rotation(b, a.inverse);
type
Returns the type of object as a string ('Rotation').
tempObject = Rotation('X', .7);
chat.print(tempObject.type); //Returns 'Rotation'
Methods
blend(b, fraction)
Returns a rotation that is the interpolation between a and b according to the specifi ed fraction (0..1)
// 10% of the way from a to b
c = a.blend(b, .1);
map(vec)
Returns a Vector that results from rotating 'vec' by this Rotation
forward = orientation.map(Vector(0.0, 0.0, -1.0));
mapAxis(axis)
Returns the specifi ed axis(0, 1, 2 == x, y, z) rotated by this Rotation
forward = orientation.mapAxis(2).negate();
power(pow)
Raises the Rotation to the specifi ed power
// b rotates 2.5 times as far as a.
b = a.power(2.5);
toString()
box.rotation = Rotation(newRotation.toString())
newRotation = currentRotation.toString()
Transform
A Transform is a geometric transformation composed of an arbitrary rotation followed by a translation.
Global Methods
Transform(various...)
Creates a new Transform(spatial transformation object). The following parameter styles are understood:
• Transform() - The identity transformation (no rotation or translation)
• Transform(translateVec) - Translate by the given Vector

• Transform(rotor) - Rotate by the given Rotation
• Transform(translateVec, rotor) - a translation of a rotation
• Transform(tfm1, tfm2) - tfm1 of tfm2 (apply tfm2, then tfm1)
• Transform(trans1, tfm2, rot3, tfm4, ...) - Any number of concatenations are allowed.
In this manner, multiple iterations of transforms are not in absolute coordinates, but are added relatively
• Transform(t1.toString()) - decoding from an encoded string.
critter.transform = Transform(critterSpot, Rotation('Y', critter.yRotation));
Properties
inverse
The inverse (opposite) of the transform.
// Relative transformation from a to b
rel = Transform(b, a.inverse);
rotation
The rotational component (Rotation) of the Transform.
// Normals only rotate.
newNorm = tfm.rotation.map(oldNorm);
translation

The translational component (Vector) of the Transform.
shadowPosition = tfm.translation.change(1.0, 0.0);
Methods
blend(Transform b, fl oat fraction)
Returns a Transform interpolated the specifi ed fraction (0.0 to 1.0) from a to b.
// 10% of the way from a to b
c = a.blend(b, 0.1);
map(vec)
Returns Vector 'vec' rotated and translated by this Transform.
forward = orientation.map(Vector(0.0, 0.0, -1.0));
power(pow)
Raises the Transform to the specifi ed power.
// b transforms 2.5 times as far as a.
b = a.power(2.5);
toString()
approach:
box.transform = Transform(newTransform.toString())
newTransform = currentTransform.toString()
Vector
Global Methods
Vector(various...)
Creates a new Vector (direction or position object). The following parameter styles are understood:
• Vector() - The zero vector <0.0, 0.0, 0.0>
• Vector(x, y, z) - The vector <x,y,z>
• Vector([x, y, z]) - The vector <x, y, z>
• Vector(v.toString()) - decoding from a string-encoded Vector
Also note that anywhere a Vector is accepted as a parameter, an Array of three numbers, [x, y, z] or [r, g, b],
will generally also be understood. (The exception is that you cannot dispatch Vector methods this way, so "v =
Vector(1, 2, 3).add([4, 5, 6]);" is valid, but "v = [1, 2, 3].add([4, 5, 6]);" is not.)
critter.position = Vector(5.0, fl oorHeight, 17.0);
Properties
[n]
[0..2] return the x, y, z or r, g, b components (read only)
verticalSpeed = velocity[1];
fogBlue = fog.color[2];
x, y, z
Returns the x, y, or z component (read only)
verticalSpeed = velocity.y;
r, g, b
Returns the r, g, or b component (read only)
fogRed = fog.color.r;
length
The length of the vector.

if (a.subtract(b).length < minDistance) { ... }
negated
The negative (opposite direction) of the vector.
b = a.negated;
normalized
The normalized (length == 1) version of the vector.
b = a.normalized;
Methods
add(B)
Returns the sum of this Vector with Vector B

position = position.add(moveVec);
subtract(Vector)
Returns this Vector minus Vector B
hereToThere = there.subtract(here);
change(index, value)
Returns a new vector with the specifi ed element (0 < index <= 2) changed. It is not possible to modify a component directly except using this method.
// Set y = fl oorHeight.
shadowPlace = position.change(1, fl oorHeight);
scale(s)
Returns a scaled vector, such that a.scale(s).length() == a.length()*s
moveVec = velocity.scale(deltaTime);
same(B)
Returns true if this Vector is equal to Vector B
if (!newPosition.same(oldPosition)) { ... }
dot(B)
Returns the dot product of this Vector with Vector B. (The dot product of vectors A and B is by defi nition the
cosine of the angle between A and B, times the lengths of both A and B.)
aimQuality =
aimVec.dot(targetPosition.subtract(myPosition).normalize())
cross(B)
upVec = rightVec.cross(forwardVec);
addScaled(B, scale)
Returns the sum of this Vector with a scaled version of Vector B.
// Same as c = a.add(b.scale(.3)); but faster
c = a.addScaled(b, .3);
blend(B, fraction)
Returns a Vector interpolated the specifi ed fraction from this Vector to Vector B.
// 50% blend gives average.
midPoint = minPoint.blend(maxPoint, .5);
approach:
tempVector = Vector(newVector.toString())
newVector = currentVector.toString()
Effects
Fog
Global Properties
fog
This is the global fog effect for the current world.
// Display current fog settings
dump(fog);
Properties
active
Specifi es whether the fog is active.

fog.active = true;
color
The color Vector of the fog.
// Bluish fog
fog.color = [.8, .8, 1];
dropOff
The fog drop-off style (0=hard; 1=linear; 2=1/Z-squared).
// Linear style
fog.dropOff = 1;
far
The distance at which objects are completely obscured by the fog.

fog.far = 100;
near
The distance at which the fog starts.
fog.near = 10;
Glare
Global Properties
glareEffect
Represents the glare SceneGroup for the camera.
// Display current glare settings
dump(glare);
Properties
active
Specifi es whether the glare effect is active (default is false)
glareEffect.active = true;
brightness
glareEffect.brightness = 1.0;
color
An array of three fl oats representing the scales of the color of the glare in RGB values. glareEffect.color = [0.5, 0.7, 1.0];
nonlinearity
glareEffect.nonlinearity = 0.5;
radius
glareEffect.radius = 16;
Sound
Global Methods
Sound(url)
Creates a new Sound in the current world from the specifi ed URL (but does not start playing the sound).
windSound = Sound("http://myDomain.com/myHomePage/wind.wav");
Properties
active
Is the sound playing?
// Pause the wind
windSound.active = false;
far
The distance beyond which the sound will not be audible.
windSound.far = 50.;
near
The distance within which the sound will always be at maximum value. For each additional multiple of minDistance, the sound loudness will decrease by half.
windSound.near = 2.5;
position
The absolute position vector of the sound in the world or null if the sound is ambient. When switching from
ambient to positional or vice versa, you must call play() to restart the sound in the new mode.
// Wind sounds coming from 'outside'
windSound.position = window.position;
repeats
How many times should the sound repeats (0 = forever). Cannot be changed while the sound is playing.
// knockSound.play() plays knock knock knock!
knockSound.repeat = 3;
URL
The URL from which the sound was created (translated to an absolute URL).
chat.print("Now playing: " + windSound.URL);
volume
The (maximum) volume of the sound, from 0 to 1.
windSound.volume = .5;

Methods
play()
Plays the sound from the begining.
windSound.play();
stop()
Stops the sound from playing.
ADOBE ATMOSPHERE **267**
User Guide
windSound.stop();
Event Handlers
CollisionEventHandler
A CollisionEventHandler is an object that can be used to specify what happens when two or more physical objects collide with each other.
Global Methods
CollisionEventHandler()
Creates a new CollisionEventHandler (which possesses onCollision and fi ltering methods).
boxEvents = CollisionEventHandler();
Local Callbacks
onCollision(PhysicalModelA, PhysicalModelB, normalRelativeVelocity)
boxEvents.onCollision = function(physicalModelA, physicalModelB, normalRelativeVelocity)
{
if ((physicalModelA.getParent() == player) ||
(physicalModelB.getParent() == player))
{
chat.print("This collision involved the Player!");
}
}
Local Methods
setFilterNoPhysicalModel()
Specifi es that the collision event handler should respond to all collisions between all physical models (that is, do
not limit the event handler to any specifi c SceneGroups).
268 APPENDIX E
boxEvents.setFilterNoPhysicalModel();
setFilterPhysicalModel(PhysicalModel)
Specifi es that the collision event handler should respond to all collisions for a single physical model.
boxEvents.setFilterPhysicalModel(boxPhysicalModel);
setFilterPhysicalModels(PhysicalModel, PhysicalModel)
Specifi es that the collision event handler should be limited to collisions between two physical models.
boxEvents.setFilterPhysicalModel(boxPhysicalModel, conePhysicalModel);
KeyEventHandler
Global Methods
KeyEventHandler()
// create a new handler
myEventHandler = KeyEventHandler();
ADOBE ATMOSPHERE **269**
User Guide
Local Callbacks
onEvent(keyEvent)
A method which defi nes what will happen when a key event occurs. A 'KeyEvent' object is returned which
possesses the key code for the key pressed, as well as several other properties.
myEventHandler = KeyEventHandler();
myEventHandler.setFilterOnKeyDown();
myEventHandler.onEvent = function(keyEvent)
{

```
chat.print("Key Code = " + keyEvent.keyCode);
}
```
The additional 'keyEvent' properties are as follows:
keyCode // the numeric value of the key pressed (see table below)
KeyDown // true for key down event, and false for key up
metaKey1Down // true if 'Shift' key was also held down during key event
metaKey2Down // true if 'Ctrl' key was also held down during key event
metaKey3Down // (currently not used)
metaKey4Down // (currently not used)
See the description for KeyEvent below for more details.

Local Methods

By default, a KeyEventHandler responds to all key events. Filtering Methods are therefore provided below to
limit the scope of key events and allow customized responses.

setFilterEvent()

Causes the handler to respond to all keyboard events.
```
myEventHandler = KeyEventHandler();
myEventHandler.setFilterEvent();
myEventHandler.onEvent = function(keyEvent)
{
chat.print("Key Code = " + keyEvent.keyCode);
}
```
setFilterOnKeyDown()

Causes the handler to respond to all key down events only.
```
myEventHandler = KeyEventHandler();
myEventHandler.setFilterOnKeyDown();
myEventHandler.onEvent = function(keyEvent)
{
chat.print("Key down key code = " + keyEvent.keyCode);
}
```
setFilterOnKeyUp()
```
myEventHandler = KeyEventHandler();
myEventHandler.setFilterOnKeyUp();
myEventHandler.onEvent = function(keyEvent)
{
chat.print("Key up key code = " + keyEvent.keyCode);
}
```
KeyEvent

Properties

keycode
the numeric value of the key pressed (see table below)
keydown
true for key down event, and false for key up
metaKey1Down
true if 'SHIFT' key was also held down during key event
metaKey2Down
true if 'CTRL' key was also held down during key event
metaKey3Down
(currently not used)
metaKey4Down
(currently not used)
MouseEventHandler

Global Methods

MouseEventHandler()

Creates a new MouseEventHandler which may be set to process all mouse events(default), or be limited to a specifi c primitive, viewpoint object, or physical model.

// specify that the main world geometry, or 'stage' will receive click events
myEventHandler = MouseEventHandler();
myEventHandler.setFilterOnClick(stageModel.getSolidObject(0));
myEventHandler.onEvent = function(mouseEvent, what)
{
chat.print("Object = " + what);
}
Local Callbacks
onEvent(mouseEvent, what)
A method which defi nes what will happen when a mouse event occurs. A 'mouseEvent' object is returned which
possesses numerous properties, and the object under the mouse is returned as the second argument 'what'.
myEventHandler = MouseEventHandler();
myEventHandler.setFilterOnClick(stageModel.getSolidObject(0));

myEventHandler.onEvent = function(mouseEvent, what)
{
chat.print("Object = " + what);
dump(mouseEvent);
}
If the above onEvent function is called as shown, a 'dump' of the mouseEvent will return the following properties, which can in turn be used to control other logic for the event:
buttonLeftDown // true or false state of the left mouse button
buttonMiddleDown // true or false state of the middle mouse button
buttonRightDown // (currently always false)
clickDown // true for mouse down event, and false for mouse up
clickEvent // acts as a 'type' identifi er, and returns true for all click events
clickUp // true for mouse up event, and false for mouse down
metaKey1Down // true if 'Shift' key was also held down during click event
metaKey2Down // true if 'Ctrl' key was also held down during click event
metaKey3Down // (currently not used)
metaKey4Down // (currently not used)
x // the x-axis pixel position of the mouse during the click event
y // the y-axis pixel position of the mouse during the click event
Local Methods
setFilterOnClick(object)
Causes the specifi ed object (Primitive, Viewpoint Object, or SceneGroup) to return an 'OnClick' mouse event.
chair = theStage.getPrimitive("chair");
myEventHandler.setFilterOnClick(chair);

setFilterOnMouseOver(object)
Causes the specifi ed object (Primitive, Viewpoint Object, or SceneGroup) to return an 'OnMouseOver' mouse
event.
chair = theStage.getPrimitive("chair");
myEventHandler.setFilterOnMouseOver(chair);
setFilterOnMouseMove(object)
Causes the specifi ed object (Primitive, Viewpoint Object, or SceneGroup) to return an 'OnMouseMove' mouse
event.
chair = theStage.getPrimitive("chair");
myEventHandler.setFilterOnMouseMove(chair);
setFilterOnMouseOut(object)
chair = theStage.getPrimitive("chair");

myEventHandler.setFilterOnMouseOut(chair);
setFilterEvent()
Specifi es that no events should be fi ltered, and all user mouse events (click, over, out, move) should return the
event object inclusive with all it's properties. Note that using this method will cause the second argument of the
onEvent callback to return 'undefi ned'.
myEventHandler.setFilterEvent();

MouseEvent
Local Callbacks
onEvent(mouseEvent, what)
```
myEventHandler = MouseEventHandler();
myEventHandler.setFilterOnClick(stageModel);
myEventHandler.onEvent = function(mouseEvent, what)
{
chat.print("Object = " + what);
dump(mouseEvent);
}
```
If the above onEvent function is called as shown, a 'dump' of the mouseEvent will return the following properties, which can in turn be used to control other logic for the event:
```
buttonLeftDown // true or false state of the left mouse button
buttonMiddleDown // true or false state of the middle mouse button
buttonRightDown // (currently always false)
clickDown // true for mouse down event, and false for mouse up
clickEvent // acts as a 'type' identifi er, and returns true for all click events
clickUp // true for mouse up event, and false for mouse down
metaKey1Down // true if 'Shift' key was also held down during click event
metaKey2Down // true if 'Ctrl' key was also held down during click event
metaKey3Down // (currently not used)
metaKey4Down // (currently not used)
x // the x-axis pixel position of the mouse during the click event
y // the y-axis pixel position of the mouse during the click event
```
OverlapEventHandler
Global Methods
OverlapEventHandler()
Creates a new OverlapEventHandler (which possesses onEnter, onLeave, and fi ltering methods).

buildingOverlap = OverlapEventHandler();
Local Callbacks
onEnter(PhysicalModelA, PhysicalModelB)
```
buildingOverlap.onEnter = function(physicalModelA, physicalModelB)
{
if ( physicalModelA.getParent().type == "Player" ||
physicalModelB.getParent().type == "Player" )
{
chat.print("The Player has entered the building.");
}
}
```
onLeave(PhysicalModelA, PhysicalModelB)
```
buildingOverlap.onLeave = function(physicalModelA, physicalModelB)
{
if ( physicalModelA.getParent().type == "Player" ||
physicalModelB.getParent().type == "Player" )
{
chat.print("The Player has left the building.");
```

}
}
Local Methods
setFilterNoPhysicalModel()
Specifi es that the overlap event handler should respond to all overlaps between all physical models (that is, do
not limit the event handler to any specifi c SceneGroups).
buildingOverlap.setFilterNoPhysicalModel();
setFilterPhysicalModel(PhysicalModel)
Specifi es that the overlap event handler should respond to all overlaps for a single physical model.
buildingOverlap.setFilterPhysicalModel(boxPhysicalModel);
setFilterPhysicalModels(PhysicalModel, PhysicalModel)
Specifi es that the overlap event handler should be limited to overlaps between two physical models.
buildingOverlap.setFilterPhysicalModel(boxPhysicalModel, conePhysicalModel);

Scene Objects
Actor
Global Properties
theActor
The global object representing the person navigating.
dump(theActor); // Display the user's properties
Properties
type
The object's type. Returns "theActor".
if (foo.type == 'theActor') { ... }
avatarURL

chat.print("Current avatar = " + theActor.avatarURL);
showAvatar
A fl ag that determines whether the avatar is rendered.
// Hide the avatar
theActor.showAvatar = false;
position
the position as a Vector object
theActor.position = Vector(10, 5, -25);
transform
the Transform representing the combined position and orientation.
theActor.transform = entryPoint.transform;
worldSpacePosition
The worldSpacePosition of the object as a Vector.
// set the worldSpacePosition of the object to a known anchor
myObject.worldSpacePosition = myAnchor3.worldSpacePosition;
worldSpaceTransform
The worldSpaceTransform is the combined worldSpacePosition and worldSpaceOrientation of the object.
// set the worldSpaceTransform of the object to a known anchor
myObject.worldSpaceTransform = myAnchor3.worldSpaceTransform;
facing
a Vector pointing the direction the Actor is facing
function moveActorForward(direction, distance)
{
actor.position = theActor.position.addScaled(actor.facing, dist);
}
worldSpaceFacing
a Vector pointing the direction the Actor is facing, in world space coordinates.

```
function moveActorForward(direction, distance)
{
actor.position = theActor.position.addScaled(actor.worldSpaceFacing, dist);
}
```

bodyYaw

The rotation angle about the Y axis in radians.

```
angle = theActor.bodyYaw;
```

headPitch

The pitch or tilt of the head relative to the body in radians.

```
pitchAngle = theActor.headPitch;
```

velocity

the Actor's velocity Vector in the World coordinate frame

```
theActor.velocity = Vector(0, 10, 10); // Jump up and forward
```

acceleration

An acceleration vector which is applied to the Actor.

```
// Apply an acceleration in the X direction (continuous).
theActor.acceleration = Vector(20, 0, 0);
```

worldSpaceVelocity

the Actor's velocity Vector in the World coordinate frame.

```
theActor.worldSpaceVelocity = Vector(0, 10, 10); // Jump up and forward
```

worldSpaceAcceleration

An acceleration vector which is applied to the Actor (in world space coordinates).

```
// Apply an acceleration in the X direction (continuous).
theActor.worldSpaceAcceleration = Vector(20, 0, 0);
```

bodySpaceVelocity

A vector that represents the velocity of the actor relative to the world.

```
if (theActor.bodySpaceVelocity.length > 20)
{
chat.print("The actor is moving fast.");
}
```

targetVelocity

A velocity that the actor tries to match when at rest. This may be set to non-zero when trying to track a moving
object.

```
theActor.targetVelocity = Vector(0.0, 0.0, 1.0);
```

targetWorldSpaceVelocity

A velocity that the actor tries to match when at rest, in world space coordinates. This may be set to non-zero
when trying to track a moving object.

```
theActor.targetWorldSpaceVelocity = Vector(0.0, 0.0, 1.0);
```

forwardSpeed

the Actor's forward velocity, facing the Actor's current direction for this rendered frame

```
theActor.forwardSpeed += 20; // Jump straight forward
```

lateralSpeed

the Actor's lateral velocity, facing the Actor's current direction for this rendered frame

```
theActor.lateralSpeed += 20; // Jump straight to the right
```

verticalSpeed

the Actor's vertical velocity, facing the Actor's current direction for this rendered frame

```
theActor.verticalSpeed += 20; // Jump straight up
```

enableVerticalThrusters

If enableVerticalThrusters is set to true, then holding down the shift key while using the up-arrow key reverses the direction of gravity. If gravity is off then the up and down arrows affect the actor's vertical velocity.

theActor.enableVerticalThrusters = true;
gravity
// Reduce gravity for this particular world.
theActor.gravity = 16;
forcedCollideSetting
If ignoreCollidePreference is set to true, the value of forcedCollideSetting determines whether collisions are
enabled
// To force collisions to be on,
// independent of the control panel setting: theActor.ignoreCollidePreference = true;
theActor.forcedCollideSetting = true;
// To force collisions to be off
// independent of the control panel setting:
theActor.ignoreCollidePreference = true;
theActor.forcedCollideSetting = false;
ignoreCollidePreference
If set true, this overrides the user's collide preference and enables collisions based on the value of forcedCollideSetting.
// To use the collide preference from the control panel:
theActor.ignoreCollidePreference = false;
// To force collisions to be on,
// independent of the control panel setting:
theActor.ignoreCollidePreference = true;
theActor.forcedCollideSetting = true;
// To force collisions to be off,
// independent of the control panel setting:
theActor.ignoreCollidePreference = true;
theActor.forcedCollideSetting = false;
forcedGravitySetting
If ignoreGravityPreference is set to true, the value of forcedGravitySetting determines whether gravity is enabled
// To force gravity to be on,
// independent of the control panel setting:
theActor.ignoreGravityPreference = true;
theActor.forcedGravitySetting = true;
// To force gravity to be off,
// independent of the control panel setting:
theActor.ignoreGravityPreference = true;
theActor.forcedGravitySetting = false;
ignoreGravityPreference
// To use the gravity preference from the control panel:
theActor.ignoreGravityPreference = false;
// To force gravity to be on, independent of the control panel setting:
theActor.ignoreGravityPreference = true;
theActor.forcedGravitySetting = true;
// To force gravity to be off, independent of the control panel setting:
theActor.ignoreGravityPreference = true;
theActor.forcedGravitySetting = false;
keyLeftRightMovesEnabled
A fl ag that determines whether to use the built in navigation scheme for shift+left-arrow or shift right-arrow.
// We wish to prevent such motion
theActor.keyLeftRightMovesEnabled = false;
keyUpDownMovesEnabled
A fl ag that determines whether to use the built in navigation scheme for shift+up-arrow or shift down-arrow.

// We wish to prevent such motion
theActor.keyUpDownMovesEnabled = false;
keyForwardBackwardMovesEnabled
A fl ag that determines whether to use the built in navigation scheme for up-arrow or down-arrow.
// We wish to prevent such motion
theActor.keyUpForwardBackwardMovesEnabled = false;
keyLeftRightTurnsEnabled
A fl ag that determines whether to use the built in navigation scheme for left-arrow or right-arrow.
// We wish to prevent such motion
theActor.keyLeftRightTurnsEnabled = false;
keyUpDownTurnsEnabled
A fl ag that determines whether to use the built in navigation scheme for ctrl+up-arrow or ctrl down-arrow.
// We wish to prevent such motion
theActor.keyUpDownTurnsEnabled = false;
mouseLeftRightMovesEnabled
A fl ag that determines whether to use the built in navigation scheme for shift+left-arrow or shift right-arrow.
// We wish to prevent such motion
theActor.mouseLeftRightMovesEnabled = false;
mouseUpDownMovesEnabled
A fl ag that determines whether to use the built in navigation scheme for shift+mouse-up or shift mouse-down.
// We wish to prevent such motion
theActor.mouseUpDownMovesEnabled = false;
mouseForwardBackwardMovesEnabled
A fl ag that determines whether to use the built in navigation scheme for mouse-up or mouse-down.
// We wish to prevent such motion
theActor.mouseForwardBackwardMovesEnabled = false;
mouseLeftRightTurnsEnabled
A fl ag that determines whether to use the built in navigation scheme for mouse-left or mouse-right or ctrl mouse-left or ctrl mouse-right.
// We wish to prevent such motion
theActor.mouseLeftRightTurnsEnabled = false;
mouseUpDownTurnsEnabled
A fl ag that determines whether to use the built in navigation scheme for ctrl+ mouse-up or ctrl+mouse-down.
// We wish to prevent such motion
theActor.mouseUpDownTurnsEnabled = false;
keyMoveInsteadOfTurn
A fl ag that determines whether to move instead of turn the theActor when the left-arrow or right-arrow is pressed. Default is false.
// Make the actor strafe
theActor.keyMoveInsteadOfTurn = true;
Methods
dist(position)
Returns the distance between the Actor and the provided position Vector.
if (theActor.dist(frog.position) < frog.fearRadius)
{

frog.jumpAwayFrom(theActor.position);
}
moveTo(Vector position)
target = stageModel.getPrimitive("Target");
theActor.moveTo(target.position);

transformTo(transform)

```
anchor = stageModel.getPrimitive("anchor");
theActor.transformTo(anchor.transform);
```

alignHeadAndBodyTo(Rotation)

Given the Rotation, aligns the body subject to the constraint that it can only rotate around the y-axis, and aligns
the head subject to the constraint that it can only rotate around a horizontal axis relative to the body.

```
// Given a subject, align the actor to look at it
deltaPos = theActor.position.subtract(subject.position);
actorRange = Math.sqrt(deltaPos.x*deltaPos.x +
deltaPos.y*deltaPos.y +
deltaPos.z*deltaPos.z);
actorRangeXZ = Math.sqrt(deltaPos.x*deltaPos.x +
deltaPos.z*deltaPos.z);
actorPitch = Math.atan2(-deltaPos.y, actorRangeXZ);
actorYaw = Math.atan2(deltaPos.x, deltaPos.z);
targetOrientation = Rotation('YX', actorYaw, actorPitch);
theActor.alignHeadAndBodyTo(targetOrientation);
```

getBodyOrientation()

Returns the Transform representing the actor's body orientation. Note that the actor's body does not tilt up and
down, only rotates around the world's y-axis.

```
// Calculate the actor's body's yaw
bodyOrientation = theActor.getBodyOrientation().map(Vector(0,0,1));
bodyYaw = Math.atan2(bodyOrientation.z, bodyOrientation.x);
```

getHeadOrientation()

The Transform representing the orientation of the actor's head.

```
// Places the camera six feet behind the actor
offset = Vector(0.0, 0.0, 6.0);
bodyRelativeOffset = theActor.getHeadOrientation().map(offset);
camera.position = theActor.position.add(bodyRelativeOffset);
camera.orientation = theActor.getHeadOrientation();
```

setPosition(Vector or x,y,z)

```
// Place the actor at the exact location specifi ed.
target = Vector(10.0, 0.0, 6.0);
theActor.setPosition(target);
// or
theActor.setPosition(target.x, target.y, target.z);
```

getPositionAvoidingCollisions(Vector or x,y,z)

```
// Find a location as close to specifi ed target as possible,
// while avoiding collisions with the fl oor, etc.
target = anchor7.position;
theActor.getPositionAvoidingCollisions(target);
// or
theActor.getPositionAvoidingCollisions(target.x, target.y, target.z);
```

setPositionAvoidingCollisions(Vector or x,y,z)

```
// Places the actor as close to specifi ed target as possible,
// while avoiding collisions with the fl oor etc.
target = Vector(10.0, 0.0, 6.0);
theActor.setPositionAvoidingCollisions(target);
// or
theActor.setPositionAvoidingCollisions(target.x, target.y, target.z);
```

getSolidObjectCount()

Returns the solid object count for the Actor's SceneGroup currently in use. This is useful for modifying avatar

properties, such as dynamic lighting.
//get the actor SceneGroup geometry, and enable dynamic lighting
for (i = 0; i < theActor.getSolidObjectCount(); i++)

{
theActor.getSolidObject(i).useDynamicLighting = true;
}
getSolidObject(index)
Returns the SolidObject at the given index that is contained by the actor's SceneGroup.
//print actor height compared to the world
currentHeight = (theActor.getSolidObject(0).position[1]
- stageModel.getSolidObject(0).position[1]);
chat.print("Your avatar is currently this high: " + currentHeight);
getViewpointObjectCount()
Returns the Viewpoint object count for the Actor's SceneGroup currently in use. This is useful for modifying
avatar properties, such as dynamic lighting.
for (i = 0; i < theActor.getViewpointObjectCount(); i++)
{
theActor.getViewpointObject(i).useDynamicLighting = true;
theActor.getViewpointObject(i).addDynamicHighlights = true;
}
getViewpointObject(index)
type
The object's type; returns "Anchor".
if (myObject.type == 'Anchor') { ... }
name
Returns the name of the object as assigned in the Application.
if (myObject.name == 'myCoolObject') { ... }
loaded
Returns true once the object has fi nished loading.

if (myObject.loaded) { ... }
parent
A reference to the parent SceneGroup containing the object
myParent = myObject.parent;
position
The position of the object as a Vector.
// set the position of the object to a known anchor
myObject.position = myAnchor3.position;
orientation
The rotational orientation of the object (as a Rotation).
// set the orientation of the object to a known anchor
myObject.orientation = myAnchor3.orientation;
transform
The Transform is the combined position and orientation of the object.
// set the transform of the object to a known anchor
myObject.transform = myAnchor3.transform;
worldSpacePosition
The worldSpacePosition of the object as a Vector.
// set the worldSpacePosition of the object to a known anchor
myObject.worldSpacePosition = myAnchor3.worldSpacePosition;
worldSpaceOrientation
The worldSpaceOrientation of the object (as a Rotation).
// set the worldSpaceOrientation of the object to a known anchor

```
myObject.worldSpaceOrientation = myAnchor3.worldSpaceOrientation;
```

worldSpaceTransform

The worldSpaceTransform is the combined worldSpacePosition and worldSpaceOrientation of the object.

```
// set the worldSpaceTransform of the object to a known anchor
myObject.worldSpaceTransform = myAnchor3.worldSpaceTransform;
```