University College London

MSc Vision, Imaging & Virtual Environments

Social Phobia

Chien-yu Lin
September 2002
Supervisor: Mel Slater

Abstract

This project attempts to find out how effective virtual reality would be used in treating social phobia. The aim of the project is to design a virtual tube train and simulate the social encounters that might happened in the tube train. We hope to find out the factors to increase the level of presence in the virtual reality so that people who took the virtual tube would get similar physiological and psychological responses as they would get from taking a real tube train. Step by step procedures taken to implement the experiment were outlined and discussed in this thesis. We measured how effective the designed virtual environment was by the level of anxiety felt during and after the experiment. The results obtained suggest that VR exposure therapy could be as effective as in vivo exposure therapy in treating social phobic patients.

Acknowledgements

A great appreciation to my supervisor Professor Mel Slater for his patience, guidance and encouragement throughout the project. Thanks to my best friend in the course who is also my project partner Laura Kayte James for her help and moral support throughout MSc. VIVE course. Thanks to Dr Anthony Steed and postgraduate student Joel Jordan for their guidance when difficulties were encountered in VRJuggler. Thanks to Dave Swapp the CAVE manager for helping us to tackle the problems encountered in the audio files. Thanks to my fellow course friends, Samuel Pang, Richard Fok and Lisa N. Gralewski for a wonderful year. Finally, thanks to my husband Steven Cheung for prove reading my thesis.

This project, Social Phobia, was a joint effort by Chien-yu Lin and Laura Kayte James. The design, implementation of the hierarchy of the avatars, and execution of the experiment was a collaborative effort. The design of the scenario and the implementation of the behaviour of the avatars are carried out individually. The research of background literature, the analysis of the experiment results and the presentation of the results in the form of a thesis were carried out individually without any consulting each other.

Contents

1. Introduction

In this chapter, a brief description of social phobia is given. It also introduces the aim; the possible scope and any contributions made to this project.

1.1 Project Aims

Social phobia is a psychological disorder characterised by a fear of social situations. There is a high prevalence of social phobia in the community and the course of the condition tends to be chronic. However, only a few seek for professional help. Such condition is long ignored and degraded the quality of the life of many individuals. Social phobia will be further discussed in chapter two.

The sessions to treat such condition need lot of human resources and capital. For example, to treat fear of height, the patient and the professional need to travel to a shopping centre (for the first stage) and progressively to the cliff. Such approach could be costly, in terms of time and human resources.

With this in mind, it would be ideal to find a way or model which can be used to simulate real-life situations without generating high costs in terms of time and resources.

Virtual Reality (VR) based therapy could be a solution to the problem stated above. It has a number of advantages over the disadvantages discussed above:

- The possibility of saving time on travelling: in vivo exposure therapy for fear of height, the therapist and the patient might need to travel to the country side that may be hours away from a city. If a VR therapy could be offered in the city, the therapist and the patient could save their time on travelling.
- The possibility of reducing monetary costs: in vivo exposure therapy for fear of taking flights, the patient and therapist have to spend money on renting the studio or taking the actual flights more than one time. It will be expensive for

government health authorities such as NHS and the patients themselves. VR based therapy is cheaper because with the present technology, a good simulation could be rendered with a reasonably cheap computer and still give the required effects.

- It can provide a more controlled environmental condition: in any outdoor therapy, the weather and the interaction with actors are uncontrollable. The VR based therapy could provide a constant weather condition such as the lighting and the controlled interaction between the patient and the avatars.
- It could provide better confidentiality: as mentioned before, the social phobia patient fears that he or she will act in a way that will be humiliating or embarrassing, therefore, they need confidentiality to feel safe and comfortable during the early stages of the therapy treatment. Whereas in VR therapy, the people that are involved are only the therapist, the patient and possibly the lab assistant, thereby reducing the possibility of the patient experiencing emotions such as embarrassment.
- It could provide safety: for an acrophobia patient, it is dangerous to have an in-vivo exposure therapy at somewhere that is high. The patient himself/herself may face the danger of falling over due to the high level of anxiety they may be experiencing. Therefore, using VR based exposure therapy could ensure the safety of the patient and the therapist.

At the basic level, this project want to find out whether Virtual Reality (VR) could cause anxiety in an individual the same way as the real world. To do this, an executable experiment with measurable dependent and independent variables has been formulated. The measured results from these experiments could allow us to accept, reject or answer the following hypothesis.

- Does sound play an important part in creating the sense of presence? Does such presence created by sound provide the stimuli (i.e. the avatars) the ability to create anxiety similar in real life?
- Does previous VR experiences will influence the sense of presence? Is the sense of presence enhanced or degraded with the amount of previous VR experience?

- Is it better to have an automatic experiment that does not need a person to issue command by keyboard or a manual experiment that allows the avatar to interact with the test subject controlled by the observer?

By answering these questions, this will hopefully contribute to the first of the overall aim of this project by developing a sophisticated therapy tool so that social phobia sufferers receive can the treatment they need in a less costly and safer environment, so giving them the opportunity to improve their quality of life.

1.2 Scope And Contribution

This project is a partly joint project. The hierarchies of the avatar's joint are created by Laura James and Chien-yu Lin with the help from a postgraduate, Vinoba Vinayagamoorthy. After successfully creating the hierarchies, individual scenario story line, avatars construction and avatars behaviours programming was carried out individually. The supervisor of this project, Mel Slater of UCL, contributed the two questionnaires that were used for the experiment..

The scope of the experiment is listed below:

- Create a scene that is similar to real life.
- Create an interesting story line that could stimulate the sense of presence for the subjects.
- Create human-like avatars with reasonable level of details.
- Create human-like gestures for the avatars.
- Create speeches for the avatars.
- Understand and revise the questionnaires to avoid confusion for subjects who participate in this experiment.
- Conduct experiments in the CAVE in the middle of August.
- Analyse the results from the experiment and give conclusion to the hypothesis mentioned in the last section.

1.3 Structure Of This Thesis

In chapter 2, background information will be given on social phobia and VR exposure therapy. It ends with three case studies on VR based exposure therapy that has been carried out in the past 5 years.

In chapter 3, an introduction of the software that was used in this project is given. It also explains the reason for choosing such software. The guidelines that were followed in design the environment are also given.

In chapter 4, a more detailed description of the purpose of this experiment is given with a detailed analysis of the questionnaires used in the experiment.

In chapter 5, the details of the procedures that were followed to carry out the experiment are described here. A short comparison is made on the actual and planned time and money expenditure.

In chapter 6, the analysis of the results obtained from the experiment is given. It also gives some observations that were made on the subjects during the experiment.

In chapter 7, a review of what had been achieved in this project is given. It also gives some suggestion on the possible future improvements.

2. Literature Review

In this chapter, a more detailed description of the causes of the social phobia is given. It also gives a brief description of the current therapies used to cure mental illnesses such as social phobia and a short discussion on the suitability of transforming the therapy into a VR based therapy. This chapter will be ended with a number of case studies about VR technology used in other phobia treatments.

2.1 What Is Social Phobia?

Social phobia is a disorder characterised by a persistent fear of situations involving exposure to unfamiliar people, unfamiliar environments or to possible scrutiny by others. Examples of such social behaviour maybe speaking in front of an audience or to individuals, using a public toilet or eating or drinking in front of people. The sufferer fears that he or she will act in a way that will be humiliating or embarrassing. As a result, the feared situations are either avoided or endured with intense anxiety or distress[1].

The criteria for the current DSM-IV diagnosis of social phobia set by the American Psychiatric Association are listed below.

- A marked and persistent fear of one or more social or performance situations in which the person is exposed to unfamiliar people or to possible scrutiny by others. The individual fears that he or she will act in a way (or show anxiety symptoms) that will be humiliating or embarrassing. Note: In children, there must be evidence of the capacity for age-appropriate social relationships with familiar people and the anxiety must occur in peer settings, not just in interaction with adults.

- Exposure to the feared social situation almost invariably provokes anxiety, which may take the form of a situationally bound or situationally predisposed panic

[1] The definition of social phobia given by Journal of Affective Disorders 50(1998) S3-S9, Social phobia: the nature of the disorder.

attack. Note: in children, the anxiety may be expressed by crying, tantrums, freezing, or shrinking from social situations with unfamiliar people.

- The person recognises that the fear is excessive or unreasonable. Note: In children, this feature may be absent.

- The feared social or performance situations are avoided or else are endured with intense anxiety or distress.

- The avoidance, anxious anticipation, or distress in the feared social or performance situations interferes significantly with the person's normal routine, occupational (academic) functioning, or social activities or relationships, or there is marked distress about having the phobia.

- In individuals aged less than 18 year, the duration is at least 6 months.

- The fear or avoidance is not due to the direct physiologic effects of a substance (e.g. a drug of abuse, a medication) or a general medical condition and it not better accounted for by another mental disorder.

- If a general medical condition or another mental disorder is present, the fear in the first criteria is unrelated to it, e.g. the fear is not of stuttering, trembling in Parkinson's disease, or exhibiting abnormal eating behaviour in Anorexia Nervosa or Bulimia Nervosa.

A social phobia patient will exhibit one or more the conditions stated above. Social phobic and agoraphobic have similar conditions. Both of them have avoidance of social situations. The main difference is, in agoraphobia; the avoidance is due to a fear of experiencing a panic attack in a situation from which escape would be difficult. I.e. the social phobic patient would be able to walk through a busy shopping centre without unreasonable anxiety where as the agoraphobic patient having problem to perform such task.

The consequences of untreated social phobia can be severe. Untreated social phobia has a considerable effect on quality of life. The social phobia sufferers experienced an inferior quality of life in the domains of work, friends, and partner. Such impairment results in social phobia sufferers having academic underachievement, inability to work, under-performance at work, and thus financial dependency. Socially and occupationally, the sufferers are at disadvantages. Thus, this condition places a burden on society as a whole. Demands from people who suffer from social phobia has increased the awareness from the healthcare authorities, professionals and the public so that suffers can receive the treatment they need and deserve to resume a full and active role in the community.

2.2 Therapies Used To Cure Phobias And Other Mental Illness

There are many different methods used for social phobia treatment such as:

- Vivo exposure-based methods.
- Cognitive-behavioural intervention.
- Social skill training.
- Relaxation techniques.

These therapies may be used exclusively or combined in sequence depending on the circumstances. In the following sections, a brief description is given of each type of therapy.

2.2.1 Vivo Exposure-Based Method

Exposure to real-life feared situations have long been acknowledged as a central component of effective fear reduction. In fact, the effective component of other techniques such as social skill training may be the exposure to feared situations that is inherent within them. Firstly, the patient and the therapist have to develop a list of anxiety-evoking situations collaboratively. Then the patient has to confront these situations from the least to most anxiety-evoking situations. Sessions will be carried out until the patient's anxiety level is vanished or diminished to a tolerable level.

The sessions to confront the stimuli could be carried out imaginarilly or in vivo. In imaginal exposure, stimuli are suggested into the patient's imagination. At this point, the therapist will ask the patient to describe the feeling when confronting the stimuli. In vivo exposure, the stimuli are real. The therapist will give assistance and withdraw the help gradually.

However, the amount of therapist involvement, time devoted to exposure per session, the spacing of practice sessions, the inclusion of imaginal exposure, and whether patients are expected to remain in the situation until anxiety is substantially reduced could differ dramatically from individuals.

It is possible to develop exposure therapy into VR-based exposure therapy to save on money and time on travelling.   For example, to treat fear of height, the sessions at later stage might take place at cliffs. Such sessions are not only expensive and waste a lot of time to travel to the destination, it also poses physical danger, e.g. the therapist or the patient may hurt themselves during the session. Having VR-based exposure therapy ensures the safety and saves time and money on travelling.

Taking another example, in the early stage, social phobic patients that interpret strangers' attention as a negative attention could be exposed to a scene such as travelling on tube in the cave. The intensity and the frequency of such stimuli (the following stare from an avatar) could be adjusted and modified gradually from least to most. Later in the stage, the patient that was getting used to the attention might be able to use real stimuli from real people. It also keeps the confidentiality of the patient.

2.2.2 Cognitive-Behavioural Therapy

Cognitive factors are more central to the development and maintenance of social phobia than is the case for other anxiety disorder. At its very base, fear of scrutiny or negative evaluation by others is a problem of the perception of other people's motives and behaviour. Therefore, interventions that address distorted thoughts and perception are especially important components of the treatment of social phobia.

In cognitive-behavioural therapy, therapists recognise the maladaptive cognition and replace them with adaptive cognition. Patients utilise imagery of anxiety-provoking situations to identify unrealistic thoughts, challenge these thoughts, and substitute more adaptive ones in their place.

Self-instructional training (SIT) with exposure is one kind of cognitive-behavioural therapy. SIT patients were trained to identify and record negative thoughts and feelings that occurred in social situations. Using imaginal rehearsal, patients and therapist developed and practices realistic thoughts designed to increase coping with the anticipatory phase of social situations, the situation itself, and the post-situation phase in which negative bias regarding performance is common among social phobic patients.

2.2.3 Social Skills Training Therapy

Many early efforts to treat social phobia were based on the presumption that patients' anxiety is related to deficit verbal social skills such as unable to have an appropriate speech content and non-verbal social skills such as avoiding eye contact. Social skill training therapy were believed to increase these behavioural skills, thus removing the underlying cause of the anxiety and increasing the probability of successful social outcomes.

Skill training therapy employs modelling, behavioural rehearsal, corrective feedback, social reinforcement, and homework assignments to teach effective social behaviour. Marzillier et al. (1976) concluded that skill training therapy had beneficial effects on patients' social lives, including increased social activities skills or social contacts. However, although the patient increased the amount of social activities he/she is engaged, Marzillier also concluded that the treatment did not result in improved social skills or reduced anxiety and had little effect on numerous other clinical measures.

Although Marzillier had made such conclusions about skills training therapy, this type of therapy is still used by many therapists in helping social phobic patients and there have been a number of successes for patients who take this type of treatment.

With this in mind, there is a great potential to develop a VR based skill training therapy. It is because the sessions for skill training therapy are time and human-resource intensive. The teachings of social behaviour can be developed to suit different models. For example, a virtual training session could be carried out on Internet web pages by formulating situations and proper behaviour answer. In this project, the tube scene can be a teaching tool for the social phobic patient to know how to handle the gestures from strangers such as eye contacts and the pub scene would be a good teaching tool to teach social phobic patient the appropriate speech contents. Again, such VR based training provides confidentiality for the patients.

2.2.4 Relaxation Techniques

The application of relaxation techniques to social phobia is based on the straightforward notion that they should provide the person with a means of coping with the physiological manifestations of anxiety, i.e. systematically desensitise the social phobic sufferer.

Patients are taught to recognise early sign of their anxiety and cope with their anxiety rather them overwhelm by it. The procedural goal of the patient is to learn to relax in 20-30 seconds and to use this skill to counteract physical symptoms encountered in problematic situations. In early sessions, patients identify their own specific early signs of anxiety through self-monitoring exercises. They are taught the skills of progressive muscle relaxation, a widely used procedure that involves the alternating tensing and relaxing of specific muscle group as the next step. As they become proficient with these procedures through repeated practice, they learn to relax without first tensing their muscles and then to relax in response to the cue word “relax”. The next step is to maintain a state of relaxation while engaged in a variety of physical activities. The final stage would be application training. It involves utilisation of relaxation skills during role-plays of anxiety-evoking social situations and during exposure to target situations that occur between sessions.

The potential of translating relaxation therapy into VR-based therapy is low. We can see that the relaxation techniques need to be taught by therapist personally to monitor the conditions.

2.3 Case Studies for VR Using In Therapy

There are a few VR based therapy studies that have been carried out. Most of them are using exposure therapy method as the implementation protocol. The following sections will give a brief description of the studies and the effectiveness of VR exposure therapy to real environment exposure.

2.3.1 Treating Fear of Driving

Driving phobia is characterised by intense, persistent fear of driving, which increases as a person anticipates, or is exposed to driving stimuli. People with driving phobia acknowledge that their fears are excessive or unreasonable, yet are unable to drive, or tolerate driving with considerable stress.[2]

The University of British Columbia carried out a study on one driving phobic patient. Each treating session consisted of a 5-10 minutes discussion of the VR scenarios to be practised in the session followed by VR exposure. Four scenarios had been constructed that was ranked by the patient from least to most anxiety provoking:

- rural residential driving
- highway driving with bridge
- residential driving with school zone
- highway driving with merging

The patient was asked to verbally report peak anxiety (100-0) at the end of each completed trial. The scenario would be repeated until the anxiety level drop to 10 or less. When the anxiety level drop to 10 or less, a more anxiety-evoking scenario will be shown.

Result shows that VR exposure for driving phobia is effective. Clinical improvement was maintained at the follow-up assessment. The patient mentioned that if it was not for VR treatment, she might not take the risk to accept the job offer. It also showed that phobia treatment does help phobic patient to have a better life and to perform the social role better.

The current drawbacks of VR exposure in treating driving phobia are limited availability and high costs (creating a physical car simulator). However, using VR exposure therapy is still worthwhile as it provides a higher level of safety to the patient and the therapist. Using a VR exposure treatment for such phobia can reduce the risk of life threatening accidents which can occur in a real-life vivo therapy treatment. In real-life vivo therapy treatments, the patient could be subjected to an elevated anxiety state, therefore reducing the capability and capacity to process information in their brain, thus impairing driving performance. This situation could expose the patient and the therapist to danger to themselves and others.

2.3.2 Treating Acrophobia

Acrophobia is known as fear of height. It is an illness that was proved that vivo exposure is more effective than relaxation strategy and imaginary exposure. Due to vivo exposure therapy is more cost-ineffective, time-consuming and the scene poses danger on the patient and therapist, VR based exposure therapy makes a good replacement of the real scene exposure.

[2] This is the definition given in the paper titled "Efficacy of Virtual Reality Exposure Therapy To Treat Driving Phobia: A Case Report", Jaye Wald, Steven Taylor, 17 April 2001.

A project was carried out in University of Amsterdam. A group of patients were exposed to three different virtual environments that had been created for this project:

- a mall in Amsterdam(Magna Plaza, approximately 40 feet high on the top floor)
- a fire escape in the centre of Amsterdam (approximately 50 feet high)
- a roof garden at top of a university building(approximately 65 feel high)

VR exposure was gradual and the therapist gave verbal guidance and encouragement to the patients. Patients were continuously prompted to look over the railing to ground floor. Then patients had to rate their anxiety level on regular time interval during the VR exposure exercise. When the anxiety was diminished to a reasonable level, therapist introduces the patient to a more difficult exercise.

Another group of patients was doing the real environment vivo exposure using the same three locations. Exposure was gradual and therapist gave verbal encouragement and guidance. If necessary, the therapist accompanied the patient but the help was withdrawn as soon as possible.

The result shows that VR exposure is as effective as real environment exposure. However, the overall anxiety level experienced during VR exposure was somewhat lower than the anxiety experienced during real environment exposure.

To conclude, VR exposure can be effective with relatively cheap hardware and software on stand-alone computers currently on the market. This suggests that VR exposure will come within reach of the ordinary practitioner within the next few years.

2.3.3 Treating Fear Of Taking Flights

Fear of flying is a common problem in modern society. In Germany, a survey showed that 15% of the population have this fear. More than 60% of passengers with fear of flying try to reduce their fear with sedatives or alcohol.
VR exposure appears to be especially suitable for the treatment of fear of flying due to the following reasons:

- Vivo exposure therapy for flying is expensive. Therefore, repeated exposure is rarely realised.
- In-vivo flying is an all-or-nothing decision (either a definite yes or no), it is impossible to have scenarios graded from least to most anxiety invoking. Therefore, patients may be reluctant to start exposure treatment.
- Individual needs are difficult to meet with in-vivo exposure because the therapist has no or only limited control over specific environmental factors such as the weather conditions, duration of the flight and the number of times of air turbulence experienced during the journey.

Therefore, VR exposure may help to circumvent those problems stated above and motivate patients for future exposure therapy.

The visual cues were presented by a head-mounted display (HMD) with an integrated earphone. The audio cues were original aeroplane sounds and flight announcement. The patients were instructed to wear the HMD and sit on the motion base that would simulate the motions of the flight such as take-off, landing and passing through air turbulence. The patients were free to look around in their seats. Patients were encouraged not to suppress anxiety during the VR treatment flight.

Results show that VR exposure was able to elicit the anxieties, and that these responses decreased within exposures and across repeated exposures. The reduction in anxiety induced by four VR exposure flights was comparable to or even greater than that induced by relaxation therapy for the same duration. To conclude, VR offers a new and promising approach for treatment of fear of flying.

3. Software Implementation

In this chapter, an introduction on the software used to construct the avatar and the scene is given. It also gives a brief description of how the virtual spaced was designed and the guidelines that were followed during the implementation stage.

3.1 Software Packages Used

To program the movement of the avatars and the hierarchy of the joints, VRJuggler was chosen as the implementation language. To construct the avatar, AC3D was chosen to implement the structures of the avatar.

3.1.1 VRJuggler

"Code Once, Experience Everywhere"
VR Juggler provides virtual reality (VR) software developers with a suite of application programming interfaces (APIs) that abstract, and hence simplify, all interface aspects of their program including the display surfaces, object tracking, selection and navigation, graphics rendering engines, and graphical user interfaces. An application written with VR Juggler is essentially independent of device, computer platform, and VR system. VR Juggler may be run with any combination of immersive technologies and computational hardware.

VR Juggler is scalable from simple desktop systems like PCs to complex multi-screen systems running on high-end workstations and super computers. The VR Juggler development environment supports many VR system configurations including desktop VR, HMD, CAVE™-like devices, and Powerwall™-like devices." [1]

VR Juggler is the open source virtual reality tool, which was developed by Iowa State University. This active research project is headed by Dr. Carolina Cruz-Neira and a team of students at Iowa State University's Virtual Reality Applications Center.

VRJuggler is an object oriented software library that uses C++ and Java as the implementation language. The main features of this library are:

- This library is portable across different platforms such as IRIX, Linux, Win32, Solaris.
- The library contains device abstractions that allows input devices such as head position tracker, wand, glove accessed through high-level interfaces. So that application does not depend on display configurations or input device specifics.
- The library has embedded performance analysis tools that the performance data collected at execution time. It is also capable to do live display of the performance data remotely.
- The library is extensible. The object-oriented design allows input devices and graphics APIs to be added easily. As mentioned before, this library is an open source. Therefore, the source code is available to public and users are free to add more functions or header files to the library.
- The library allows dynamic reconfiguration. Therefore, devices and displays can be added, removed and reconfigured without restarting the application.

VRJuggler applications are treated as objects. All applications implement a common interface then the object is handed to kernel that manages execution. Therefore, it allows switching between applications i.e. it can have multiple simultaneous applications.

Most of the existing projects for CAVE application is based on DIVE. To move to a new paradigm, an object oriented implementing language is preferred. VRJuggler is chosen because it is object-oriented and it has the benefits mentioned above. As the author does not have much experience in programming in C++, few weeks has been allocated to get familiar with C++ programming language.

3.1.2 AC3D

AC3D is a graphic software that allows user to create objects by click-and-drag motion that was used by many educational bodies such as University College London (UCL) and is also used by commercial industries such as Volkswagen for designing their automobiles. AC3D's popularity is due to the following features:

- It is cheap compare to other 3D-studio software. E.g. 3D-studio max cost around £600 GBP and AC3D only costs $39.99 USD which is within many students' financial range.
- It has a user-friendly interface.
- It uses built-in fast OpenGL 3D renderer with adjustable field-of-view so that the user can view the results immediately in 3D.
- The created object could be edited at different levels such as groups, individual objects and at individual vertices.
- It supports texture mapping.
- Its colour palette allows diffuse, ambient, emissive, specular, shininess and transparency to be adjusted to suit the object material.
- It is platform portable. It supports different operating system such as Linux, SOLARIS, and Win32.
- It can output the created model in multiple formats such as triangles, DIVE, VRML, 3D-studio.
- Is allows extrusion to make 2D lines into 3D shapes.
- It allows optimisation to remove duplicated vertices and surfaces. It helps to reduce the number of triangles the SGI machines needs to render.
- It also allows extension. It comes with a software development kit that allows plug-in interface with source code for existing plug-ins to be updated and coded for specific individual use.

AC3D was chosen as the graphic design tool because it is easily and cheaply available. Furthermore, some experience with this software has been encountered previously from a mini project. All drawn objects in this project will be outputted in triangle format. E.g. the three 3D co-ordinates and the colour coding in hexadecimal.

3.2 The Design Of Virtual Space

The chosen background environment was the trains on the London underground service, or locally known as "the tube". It was created using AC3D. They were encoded in triangle format, which was compatible with VRJuggler.

Implementation Guidelines For Tube Train:

The size and the shape of the tube train were decided to follow the criteria given below:

- The interior should be similar to the tube train we experienced everyday. I.e. limited walking area with avatars inside. The tube design implemented was based on the tube trains that runs on the Piccadilly line in London.
- To prevent subjects moving wildly in the cave, an obstacle needs to be introduced in the scene to minimise the movement.
- The design of the tube should be as simple as possible to limit the number of polygons it needs to be rendered.

The implemented interior and exterior of the tube train are shown below in diagrams 1 and 2.

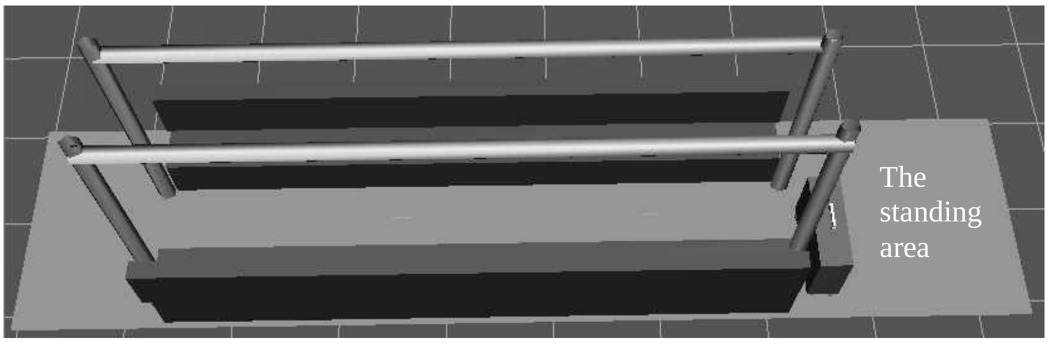

Diagram 1: The interior of the virtual tube.

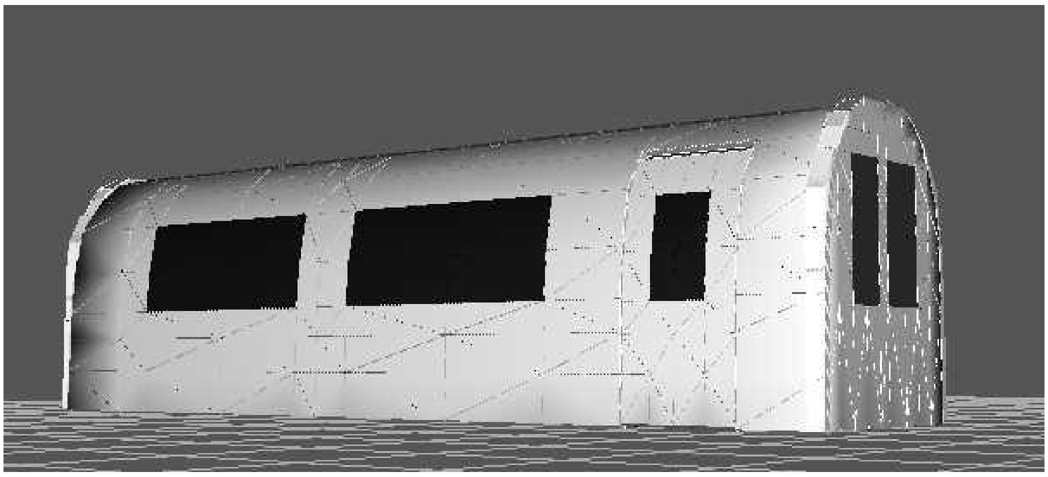

Diagram 2: The exterior of the virtual tube.

3.3 The Avatars

The joints such as knees and segments such as arms of the avatars are drawn by AC3D around the origin. Then each joint with its related segment was translated to its proper co-ordinates. The hierarchies of the joints are implemented by using the C++ programming language in VRJuggler. The movements of the avatar, which consists of rotating and translating the joints, are automatic; i.e. it does not need a lab assistant to control it.

Implementation Guidelines For Avatars:

When designing the avatars, the following criteria had been followed.

- The avatars have the minimum number of joints recommended by Humanoid Animation Work Group (HANIM).
- There is at least one human-like gesture implemented such as breathing.
- The complexity of the movement of the avatars had to be low enough that it will not slow down the execution time.
- The level of details of the avatars should be reasonably low enough that it will not slow down the frame rate.

- The behaviour of the avatars should be model like real commuters on the tube train. The behaviour of the avatars should not be extreme. E.g. shouting at the social phobic patients.

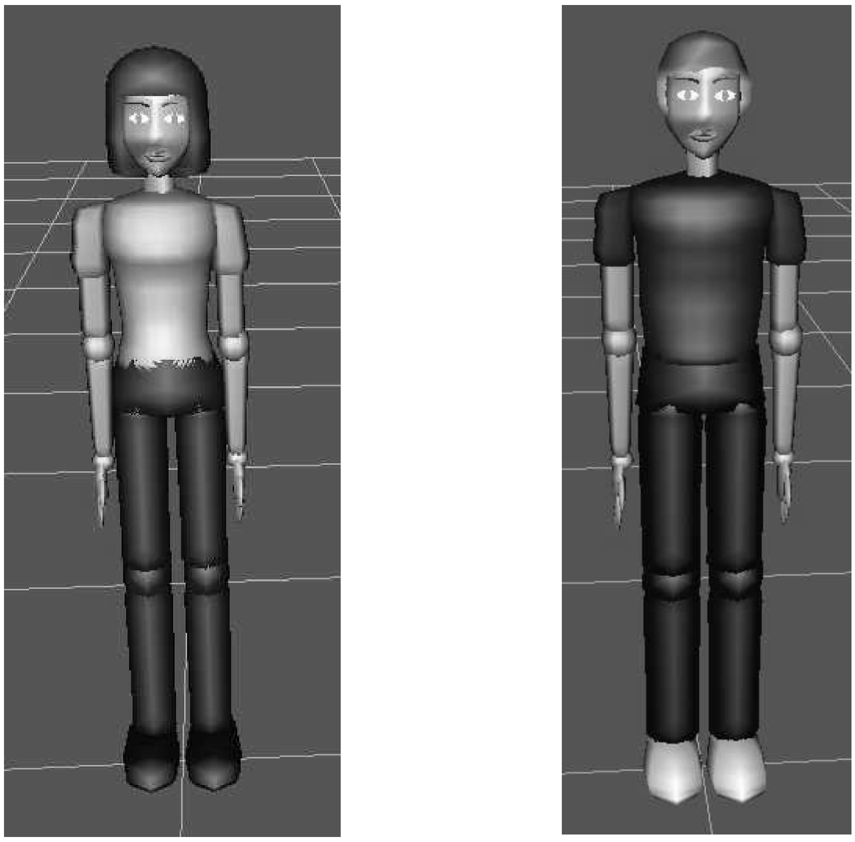

Diagram 3: The Female (left) and Male (right) Avatars.

Avatar Speech and Background Sound:

The sound files have been recorded using Windows 98 Sound Recorder using a microphone. When the subjects moves towards the seating area in the tube, this triggers one of the avatar named ZOE to prompt a question to prevent the subject from trying to sit down.

However, Sound Recorder is not a professional sound recording application. Therefore, there is some hissing in the background of the recorded speech.

In the scenario, there is no background sound. There is only the avatar speech and the broadcast message. This is due to that the SGI machine is unable to execute multiple sound files at the same time on one machine. The execution of multiple sounds files could also cause a distortion of the sound and therefore could break the sense of presence to the test subject.

4. Purpose Of The Experiment

In this project, we want to find out whether VR could trigger the subjects' feelings and emotions like the real environment. Using the tube train model, it is intended to observe what factors would make the avatars feel "alive" to the test subject and what factors contribute to the feeling of presence in the subject.

In this chapter, the design of the VR experience will be described followed by a short description of how the subjects were recruited. Then this chapter will be concluded with an analysis of the questionnaires (questionnaire A and Questionnaire B) taken with the test subject after they had taken part in the experiment.

4.1 The VR Experience

The place where the VR experience is to be held is on the tube train that operates on the Piccadilly line in central London. The subjects are free to move in the cave, which is 3 metres by 2 metre by 4 metres in size but need not to move around in the VR. Therefore, there is no practice given to the subjects to practice on navigating around in the Virtual Tube.

When the scene is loaded up, the subject will be in the standing area at the end of the tube. There is a female avatar called ZOE standing near the door on the far side of the tube train. To prevent the subject trying to sit on the seat in the virtual tube, when the subject tries to pass the male avatar called TIM to get to the seating area, ZOE will look at the subject and ask a question. At the same time, there is a big suitcase placed on the isle to give the subject a visual obstacle in his/her way to the seating area of the virtual tube. TIM, who sits near the door will constantly look at the object and follows his/her movements. BOB, the other male avatar is nodding to sleep. Other avatars is modelled to have some commuter-like behaviours such as reading a book or staring at the floor.

4.2 What We Want To Find Out

Two questionnaires were developed to find out from the test subjects their experience in the experiment. The first questionnaire; self-evaluation questionnaire, was used to find out whether the subjects have social phobia. The second questionnaire; self-evaluation questionnaire B, was used to find out how much presence the subject felt in the Virtual Environment and whether the Virtual Tube triggered any increase in the feeling of Social Phobia.

For this experiment, we want to find the answers for the following specific questions:

- Does background sound play an important part for the subject to feel the presence?
- Is it better to have automated or controlled behaviours of the avatars?
- Are there any other factors to affect the sense of presence?

A more detailed description of the two questionnaires can be found in sections 4.4 and 4.5.

4.3 How The Subjects Were Recruited

E-mails were sent at the 20th of August 2002 to groups of students within the Computer Science Department of UCL asking whether students would be interested in participating in an experiment involving VR environments . The actual content of the e-mail is supplied in the Appendix.

There was no strict restrictions for selecting the subjects. All the subjects who participate in this study are volunteers, therefore, there is no cash reward was given. All of the subjects are conformed the following criteria:

- None of the subjects know anything about the nature of the experiment.
- Not all the subjects are having social phobia.

- Having previous VR related experience is not a requirement.

5 subjects were recruited for the first experiment slot, which took place at CAVE on the 22nd of August from 9:30 a.m. to 12:00 p.m., and the first subject was treated as the pilot. As mentioned before, this is a partly joined project. All test subjects went through 2 scenarios, the virtual tube and the virtual pub. The time needed for each subject to complete a complete study is approximately half an hour. The order of displaying the scenario is at random. All subjects turned up punctually.

5 subjects were recruited for the second experiment slot, which took place at CAVE on the 24th of August, 2002 from 10:00 a.m. to 1:00 p.m. All subjects were punctual.

In total, 10 subjects were recruited which was only half of our target. However, due to the time constraints, the time given only allowed the use of 10 subjects. The list of subjects who participated in the experiment is given below:

Date	Id	Tube/Pub	Female	Male	Notes
Tue, 22.8.2002, 10:20 a.m.	1	x		x	Used as test pilot.
Tue, 22.8.2002, 11:00 a.m.	2		x		
Tue, 22.8.2002, 11:30 a.m.	3		x		No previous VR experience.
Tue, 22.8.2002, 12:00 p.m.	4	x		x	A frequent player of video game.
Tue, 22.8.2002, 12:30 p.m.	5	x	x		Very little previous VR experience.
Thurs, 24.8.2002, 10:00 a.m.	6			x	
Thurs, 24.8.2002, 10:00 a.m.	7	x	x		
Thurs, 24.8.2002, 10:00 a.m.	8		x		No previous VR experience.

Thurs, 24.8.2002, 10:00 a.m.	9	x		x	
Thurs, 24.8.2002, 10:00 a.m.	10			x	No previous VR experience.

4.4 Questionnaire A

Questionnaire A is a general self-assessment questionnaire used to find out the background of the test subjects such as their gender, status, any previous VR experience and any traits in social phobia. The first few questions used scales from 1 to 7 to rate their VR related experiences. The scale 1 indicated that the subject had never/not at all experience anything VR related whereas 7 is the opposite. The last question on the questionnaire was divided into small yes or no questions to find out how many traits they have in social phobia before running the experiment. The following table indicates what information can be obtained from questionnaire A:

QUESTION NUMBER	**INFORMATION OBTAINED**
A1	The gender of the subject.(1~2)
A2	The social status of the subject.(1~8)
A3	Any previous VR related experience.(1~7)
A4	The level of computer literacy of the subject.(1~7)
	The following question is either a **yes** or a **no**.
A5.1	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.2	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.3	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.4	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.

A5.5	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.6	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.7	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.8	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.9	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.10	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.11	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.12	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.13	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.14	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.15	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.16	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.17	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.18	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.19	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.20	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.

A5.21	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.22	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.23	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.24	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.25	If the answer is **no**, the subject has experienced social anxiety distress under this social situation.
A5.26	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.
A5.27	If the answer is **yes**, the subject has experienced social anxiety distress under this social situation.

An example of questionnaire A can be found in the appendix section.

For each sub-question in question 5, if the subject experienced social anxiety distress, the subject got a score of 1. The Social Anxiety Distress level was calculated by adding up the individual scores of the sub-questions in question 5 of the questionnaire.

4.5 Questionnaire B

Questionnaire B is also a self-assessed questionnaire. The purpose of questionnaire B was to find out from the test subject after they had completed the experiment their responses to the following:

- The level of presence the test subject felt, where a lower value indicates the subject felt a lower level of presence and high value indicates the subject felt a high level of presence in the immersive VR.

- The level of pleasantness the test subject felt, where a lower value indicates the subject felt unpleasant and a high value indicates the subject was happy in the immersive VR.
- Any physiological signs observed, therefore showing the subject felt anxious in the immersive VR.
- The social anxiety distress level that the subject may experience after the experiment.
- The level of liveliness of the avatars, where a lower value indicates the subject felt that the avatar that were present in the VR was not human-like and a higher value indicates the subject felt that the avatar is human-like.

An example of questionnaire B can be found in the appendix section.

The following table gives an analysis of the questionnaire B, indicating what we can find out from the questionnaire:

Question number	Description	Range	What it measures
1	How comfortable and relaxed with the avatars	1~7	Social phobia
2	How satisfied the subject is about his/her response to the avatar.	1~7	Social phobia
3	Sense of people in front of you	1~7	Presence
4	Sense of talking to real people	1~7	Presence and avatar liveliness
5	How human-like the avatars talking is	1~7	Presence and avatar liveliness
6	How aware of the other people	1~7	Presence
7	The impression the subject think he/she made to a person	1~7	Social phobia
8	How friendly the avatars are	1~7	Avatar liveliness and social phobia
9	How interesting the conversation is	1~7	Avatar liveliness

10	How much would the subject	1~7	Pleasantness
11	Self-rate their social response to the situation	0~100	Social phobia
12.1 ~ 12.9	Any physiological sign showing that the subject was experiencing anxiety	Yes/No	Social phobia

	Range	What it measures
13.1	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.2	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.3	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.4	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.5	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.6	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.7	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.8	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.9	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.10	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.11	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.12	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.13	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.

13.14	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.15	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.16	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.17	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.18	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.19	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.20	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.21	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.22	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.23	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.24	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.25	Yes/No	If the answer is **yes**, the subject has experienced social anxiety distress.
13.26	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.
13.27	Yes/No	If the answer is **no**, the subject has experienced social anxiety distress.

	Description	Range	What is measures
14	What were the best things about the encounter with avatars	Text	Social phobia and VR experience
15	What were the worst things about the encounter with avatars	Text	Social phobia and VR experience

16	Any other comments	Text	Social phobia and VR experience

For each sub-question in question 12 and 13, if the subject experiences social anxiety distress, the subject got 1 score. The Social Anxiety Distress level was calculated by adding up the score of question 13.

The main purpose of the experiment was to find out whether the subjects would have a feeling of presence during the experiment. If the subject has a severe social phobia and feels presence, he/she should feel an increase in social anxiety distress levels during the ride in the virtual tube.

4.6 Factors That Could Affect The Experiment.

There are a number of factors which may affect the results of the experiment. These are as follows:

- The size of the testing sample. Due to time constraints, the size of the test subjects gathered is relatively small, and therefore may not provide a good representative of people's reactions, such as social phobia, during and after the experiment. Therefore it may be possible that we would not see a increase, if any, in Social Anxiety Distress Levels after the immersive virtual tube train ride.
- The subjects chosen for the experiment may experience anxiety or distress while they are immersed in the VR experiment, but they may choose to deny such feelings, therefore affecting validity of the results.
- The subjects chosen for the experiment may not feel enough presence to trigger their emotional and physiological response i.e. VR environment created does not reflect a real-life environment.

5. Experiment Description

In this chapter, a detailed description of the scenario is given. This is followed by a description of the step-by-step procedures used to conduct the experiment.

5.1 The Scenario Description

In the experiment, there is only one scenario, a neutral scenario. The subject will be asked to take the virtual tube from Manor House underground station to Finsbury Park underground station. During this one-minute journey, the subjects are free to move around. The male avatar TIM will look at the subject and follow the subject's movement. When the subject makes an attempt to go pass TIM, the female avatar ZOE will ask the subject: "Excuse me, how to go to China Town?". At the end of the journey, there will be a broadcast message to inform the subject that the train has arrived on the platform of Finsbury Park underground station. Then the virtual tube programme is terminated. The diagram below shows an over head view which the subject will see when they enter the CAVE.

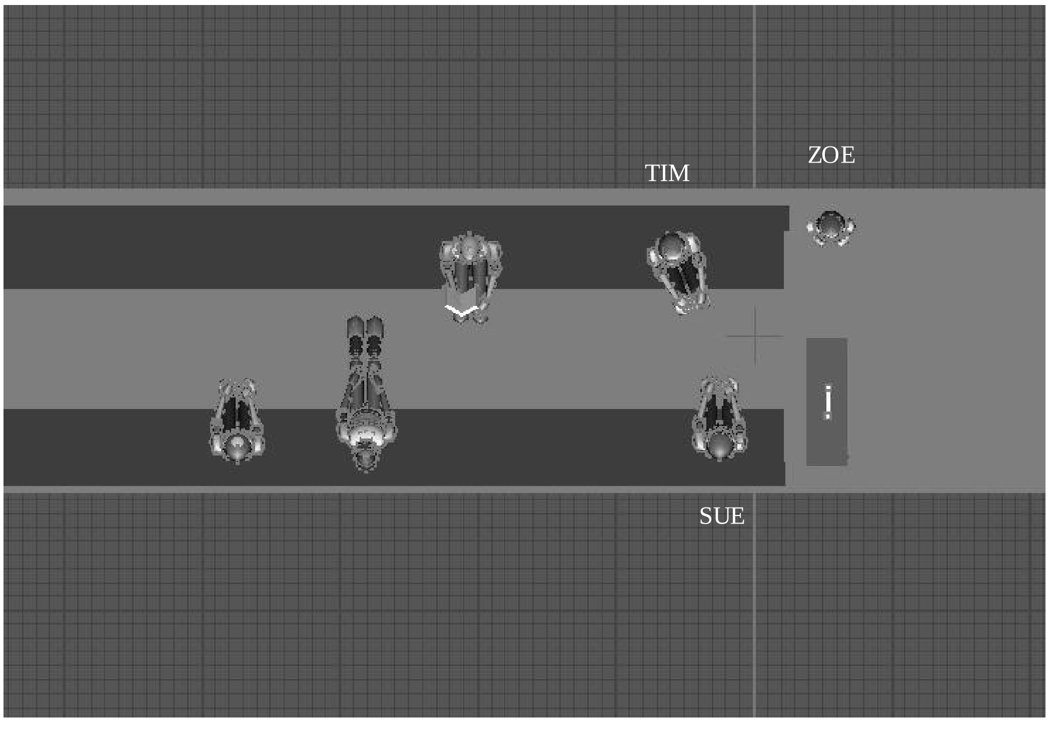

Diagram 4: Overhead View Of Tube Train Environment

5.2 Planned And Actual Expenditure Of Time And Resources

Due to there were a number of occasions where the departmental server went down, difficulties have been experienced in logging in and accessing the files. The programming for the avatars' behaviour was reliant on university computers being operational. Therefore, there was a serious delay on the progress which added to the uncertainties of whether the project will complete in time. As it approached mid-August, the availability of CAVE was low and due to the departmental server was down several times further, this pushed the availability of the CAVE to its limit. The CAVE was made available for the tube train experiment on the 20th and the 22nd of August 2002. The table below was the planned schedule made at the end of May:

DATE	**TASK DESCRIPTION**
Before 3.6.2002	Background reading
3.6.2002~16.6.2002	Get familiar with C++
17.6.2002~7.7.2002	Construct the hierarchy with Laura James
8.7.2002~14.7.2002	Construct the body parts for avatars and the tube.
15.7.2002~4.8.2002	Programming on the behaviour of the avatars.
5.8.2002~12.8.2002	Test the programme in CAVE
13.8.2002~20.8.2002	Carry out the experiment
21.8.2002~6.9.2002	Writing up the report.

However, the actual time spent is as follows:

Date	Task description
Before 3.6.2002	Background reading
3.6.2002~16.6.2002	Get familiar with C++
17.6.2002~7.7.2002	Construct the hierarchy with Laura James
8.7.2002~23.7.2002	Construct the body parts for avatars and the tube
24.7.2002~16.8.2002	Programming on the behaviour of avatars

20.8.2002~22.8.2002	Carry out experiment with 10 subjects
After 23.8.2002	Writing up report

The delay of the planned schedule is due to the following reason:

- Between the end of July till mid August, the departmental server was down. The author could not log in to access, debug and compile the files.
- The complexity of constructing the body parts of the avatar was underestimated. Each body part was constructed by dragging the vertices of the sphere to their proper position, which is very time consuming.
- Failure to install VRJuggler at home laptop. This restricted the time spent on programming avatars' behaviour and testing. At the same time, the programming depended on departmental workstations being in operational.

The planned expenditure is £0.00 as the laboratory space, workstations and required software were readily available and subjects were not paid for participating the experiment. However, to construct the body parts of the avatars in a more efficient manner, the private licence of AC3D was purchased at $39.99 USD. That brings the total expenditure on this project to £28 approximately.

5.3 Procedures And Instruction Carried Out

To make sure that the experimental set-up was uniform to each subject i.e. to avoid contamination, a precise and uniform routine was carried out for every subject before, during and after the experiment.

When the subject arrived in the meeting room outside the Cave a short greeting was given to welcome and thank the subject for their participation in the experiment. The subject was asked to sit down on the chair provided to ensure he/she was comfortable before the questionnaires were to be completed.

An instruction sheet was handed out to the subject. After the subject had finished reading the instruction sheet (which took approximately 4 minutes), a consent form was handed out for subject to complete and sign (which took approximately a further 3 minutes). When the subject had completed the consent form, questionnaire A was given to the subject to be completed. At the same time, the filled-in consent form was checked to ensure that every question was answered. This took approximately 5 minutes for the subject to complete questionnaire A. Then the subject was lead to the CAVE.

The subject was asked to remove their shoes before the start of the experiment. If the subject has had no previous experience in CAVE, the explanation of the functionality of the head tracker was given. Then verbal instructions was given to the subject depending on the scenarios. There were two scenarios which were shown to the test subjects. These are as follows:

Scenario A: The subject is on their way to the pub to meet his/her friends. He/she needs to take the tube from Manor House Station to Finsbury Park Station. During the one-and -half-minute train journey, the subject is free to move around in the CAVE.

Scenario B: The subject is on their way home after meeting his/her friend in the pub. He/she needs to take the tube from Manor House Station to Finsbury Park Station. During the one-and -half-minute training journey, the subject is free to move around in the CAVE.

Out of the 10 subjects recruited for the experiment, 5 of the test subjects were shown scenario A, and the remaining 5 test subjects were shown scenario B.

The Virtual Tube scenario is loaded up into the CAVE. After the subject had completed the tube ride, questionnaire B was given to the subject to complete. This took a further 5~6 minutes for the subject to complete. An observation was made that out of 10 subjects, 7 subjects approached lab assistant at least one time to clarify their questions in filling in questionnaire B. The most frequent asked questions are question 4 and 6. This is due to the word “people”, which was used in questions 4 and 6. The subjects were confused as to whether "the people" used in question 4 and question 6 referred to the lab assistants or the avatars.

Such confusion should be avoided in the future by reviewing the questions before the experiment is carried on further. One way to do would be to have the questions reviewed by someone who has no previous knowledge or involvement in the experiment.

After the completion of questionnaire B, the subject was guided to the sitting room outside the CAVE. A short interview was given by the lab assistants, Laura James and Chien-yu Lin. The interview took approximately 8 minutes. To ensure uniformity, each subject was asked with the same questions and no other people other than the two lab assistants and the subject himself/herself were present in the room.

The questions asked by the author was listed below:

- Do you feel that there was anyone staring at you during the journey?
- If yes, how do you feel about it?
- If no, any other comments?

The subject was then thanked for participating and ensured that any impact from the experiment would diminish to the minimum level.

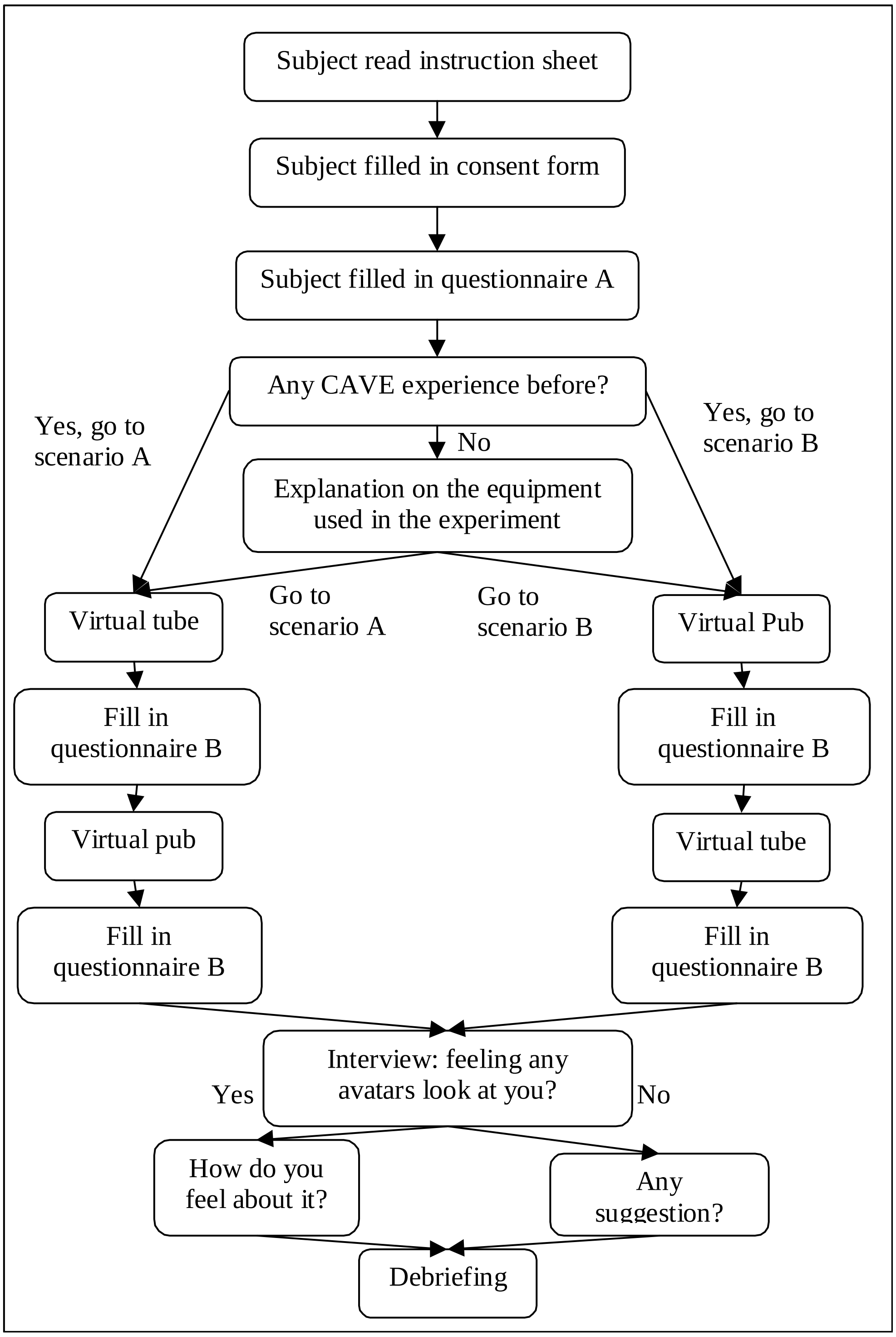

Diagram 5: Flow Diagram Of The Activities Carried During The Experiment

6. Results

After conducting the experiment, data was collected to find out what conclusions we can draw from the experiment. In this chapter, a brief introduction is given on how the data was collected, followed by the interpretation on the data collected.

6.1 Collecting Data From The Experiment

The data was collected in three ways:

1) by self-assessment questionnaires (questionnaire A, questionnaire B)
2) by recording the interview with each subject after the experiment and
3) by the observations that were made by the lab assistants during the experiment.

The last two methods of data collection were not quantifiable so they were not suitable for statistical analysis. However, they provided some interesting points for discussion.

6.2 Results

The statistical results were calculated using the data collected from questionnaire A and questionnaire B. The quantifiable parameters are Gender, Occupation, Presence, Avatar liveliness, and Pre Social Anxiety Distress level (preSAD) and post Social Anxiety Distress level (postSAD).

The statistics shown in the later sections was based on the tube scene only. Please refer to Laura James' report for the statistics for the pub scene.

6.2.1 PreSAD vs. PostSAD

To see whether the VR experience has triggered any social phobia traits on the subjects, we need to find the association between preSAD and postSAD. To do this, the preSAD level from question 5 of questionnaire A was calculated and the postSAD level from question 13 of questionnaire B was calculated so that they can be compared. The graphs that were plotted by preSAD against postSAD were plotted for scenario A and scenario B.

In scenario A, the first 5 subjects took the virtual tube followed by the virtual pub. The graph below shows the preSAD and postSAD results of the 5 test subjects who took part in scenario A.

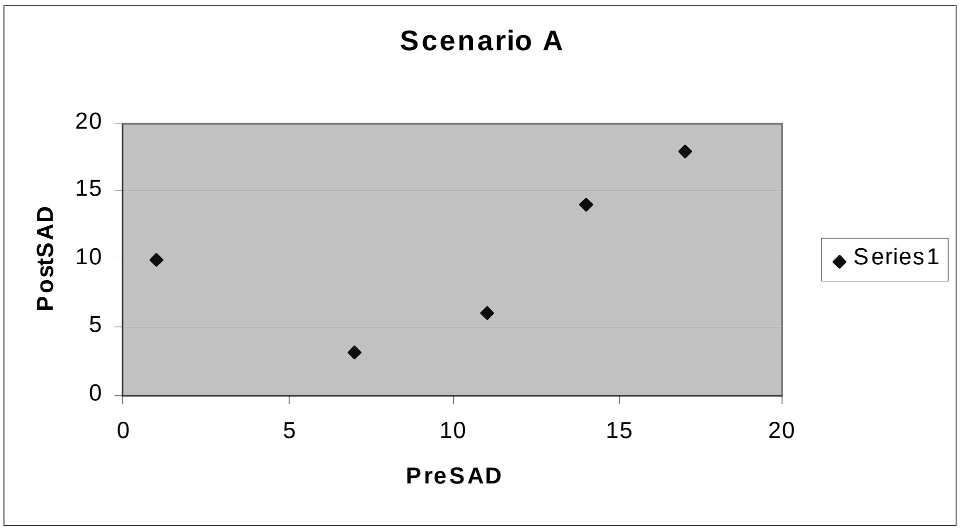

Diagram 6: PreSAD and PostSAD Results For Scenario A

In scenario B, the 5 subjects went to the pub then took the tube.

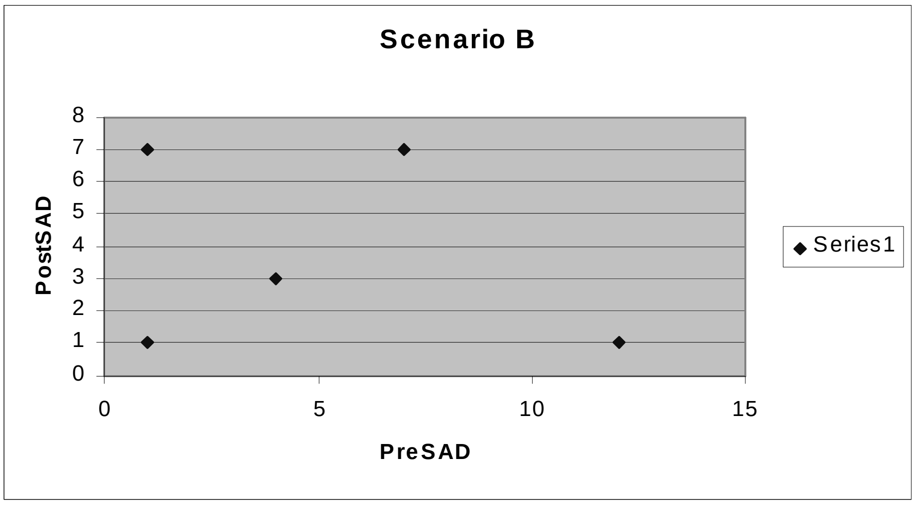

Diagram 7: PreSAD and PostSAD Results For Scenario B

We can observe that the graph in scenario A is nearly a straight line whereas in scenario B, the results are distributed at random. This could be due to the fact that virtual tube is a fairly neutral scene that does not trigger anxieties unless the subject has a severe social phobia. Therefore, after going through the pub scene, which has more social phobic stimuli, the post influence had effects on the result obtained for scenario B.

6.2.2 Presence vs. PostSAD

To find out whether high presence does trigger a higher level of anxiety, we need to find out the correlation of presence and post SAD. The level of presence was calculated by summing up question 3 to question 6 in questionnaire B.

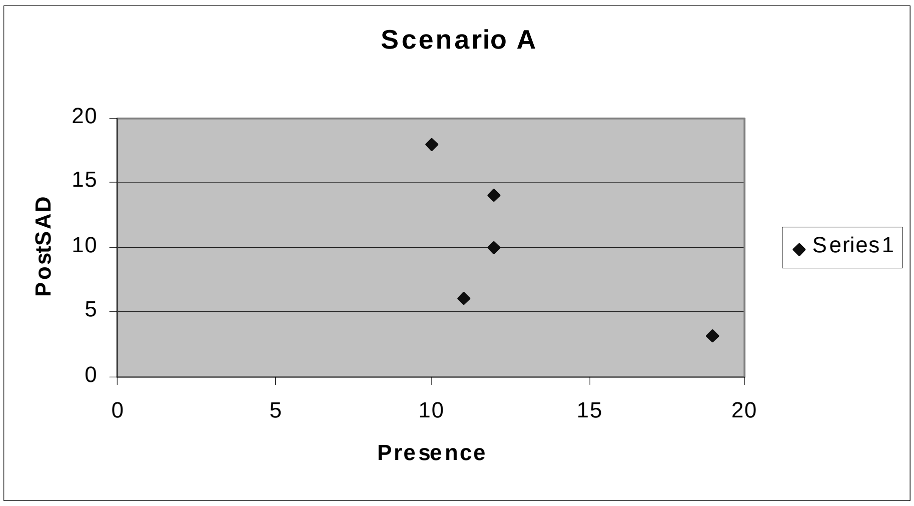

Diagram 8: PostSAD and Presence Correlation Results For Scenario A

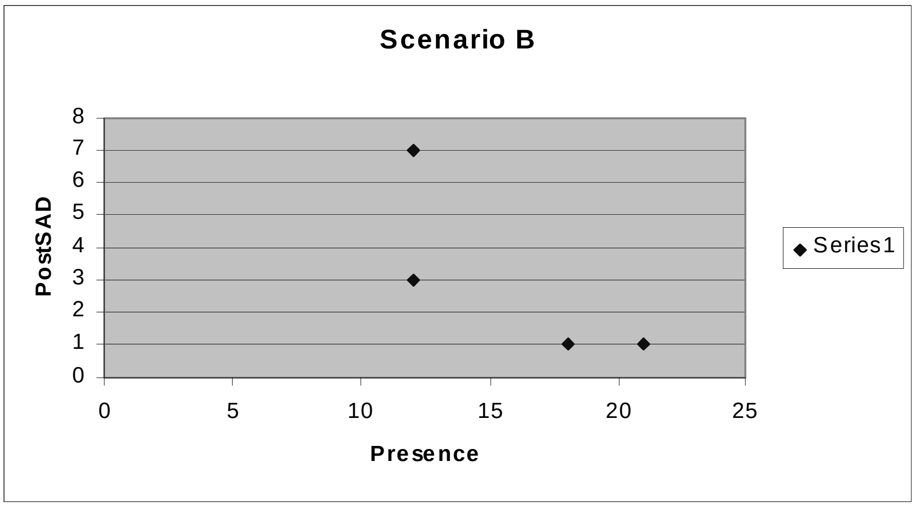

Diagram 9: PostSAD and Presence Correlation Results For Scenario B

From the two graphs above, we can observe that in scenario A, subjects do feel anxiety as the anxiety level is high (range from 5 to 20). In scenario B, subjects do not feel anxiety as much as scenario A (range from 1 to 7). There are two possibilities that contribute to such observation. The first possibility is there is a contamination between the two scenes. As mentioned before, the pub scene is more social phobia triggering. After going through the pub scene, the mild stimuli in the tube scene lose their effects. Therefore, there is a great decrease in average postSAD level compared to scenario A. The second possibility is that after the subject had been through the pub scene that gave him/her more experience with the VR, he/she felt more comfortable with VR. Therefore, a great reduction of the level of anxiety was observed.

From the graphs, it also suggests that the higher the presence, the more comfortable the subject felt. This might be because the tube scene was modelled according to the real-life situation. Subjects with high level of presence associated themselves as they were travelling on a real tube; it is just another normal tube ride for them. Therefore, they felt calm in the tube scene.

6.2.3 Previous VR experience vs. Presence

To see whether having previous experience with VR application will affect the sense of presence for the subject, we need to find out whether there is a correlation between the level of presence and their experience with VR application for the subjects. Subjects' level of VR experience could be found in question 3 in questionnaire A and the relative level of presence was found by summing up question 3 to question 6 in questionnaire B.

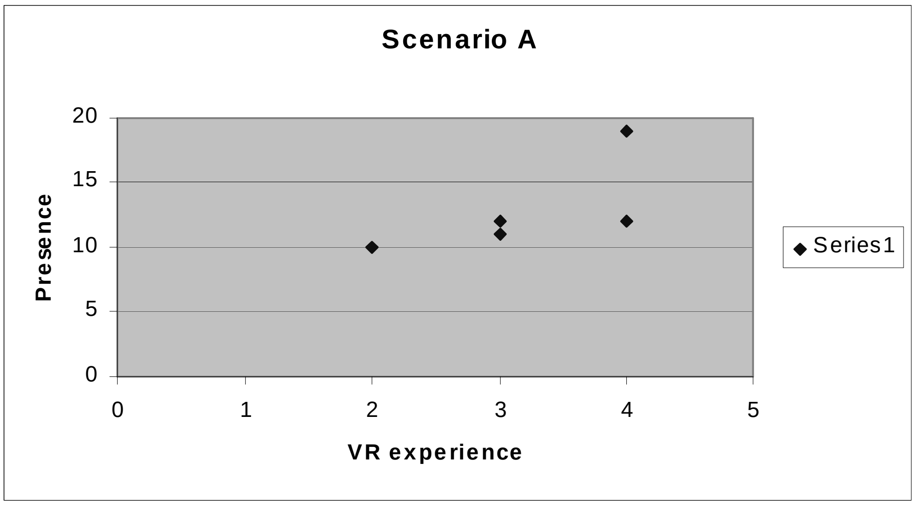

Diagram 10: The Correlation Between Presence and VR Experience For Scenario A

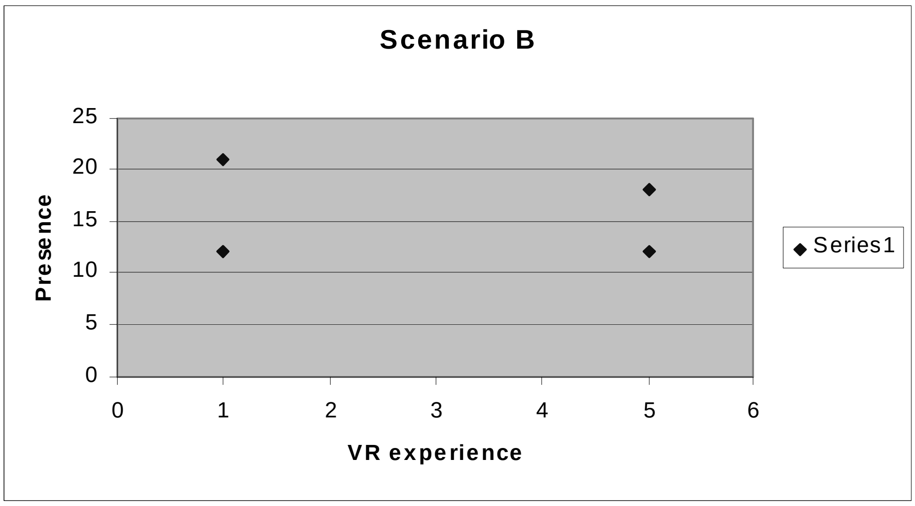

Diagram 11: The Correlation Between Presence and VR Experience For Scenario B

From the graph plotted for scenario A, we can see that the previous experience does enhance the sense of presence. The higher level of the VR experience, the higher level of presence the subject felt. However, in scenario B, there is no pattern to follow. This is due to the pub scene gave the subject some experiences in VR. Therefore, there is a contamination for the data collected for scenario B.

Due to the randomness of assigning subjects to scenario A or scenario B, the comparison between male and female could not be carried out because the condition is not uniform. However, when comparing the statistics for question 11 in questionnaire B, the self-rating social performance in the tube scene appears to be better than the pub scene. This is due to the mild stimuli which made the subjects feel more comfortable so the subject felt satisfied about their own responses.

6.3 Discussion

There are some observations that are worth discussing to improve the design of the VR application in the future.

- Out of 10 subjects, 10 subjects found it difficult to find out which avatar was speaking to them. It is because there is no mouth movement and the subjects need the mouth movement to find out which avatar was speaking. Another factor which also contributed to the confusion was the avatar speech played in the CAVE does not give any direction information i.e. where the sound was coming from.

- The mild stimuli, the eye contact, made the subjects feel that the avatars were alive. E.g. subject number 2 mentioned in the interview that the attention from the avatars made her felt that the avatars are alive. No one noticed that the avatars breathed. However, the human-like gesture such as reading a book made the avatars more human-like.
- Most of the subjects felt that the tube scene is more comfortable. This is because in the tube scene, the subjects were not pressured to make conversation.
- The position of the head tracker placed in the CAVE is important. When the pilot was doing the experiment, the head tracker was placed at the centre of the CAVE. The pilot was standing still in the centre throughout the whole experiment, which was not desired and not following the plot. Instead, the head tracker was placed to the left of the CAVE, therefore pressuring the subjects to move around in the CAVE i.e. move from the left of the CAVE to the centre, and hence gave a better sense of presence to them.
- The use of automated avatars allowed the subjects feel presence without being distracted by the click sound from the keyboard and the embarrassment made by the lab assistants observing them. However, automated avatars' machine-like replies and simple reactions degraded the sense of presence at the same time. E.g. subject number 9 mentioned in the interview that the repeated speech makes the avatars machine-like.
- It was also observed that background sound is important to help the subjects to feel presence. Subject number 3 mentioned that she felt that the background sound is missing hence the sense of presence is reduced.
- Subjects with no or little previous experience with VR did try to avoid the obstacle presented in the VR. E.g. subject number 2 tried to avoid the suitcase that was placed in the middle of the aisle.
- The scenario design is important. One of the subjects mentioned during the interview that when she tried to find a place to sit on the virtual tube like she

would in the real tube, she found that she could not sit on the virtual tube. She lost the sense of presence at that moment.

- Many subjects mentioned that after they showed the avatar ZOE the way to China Town, they expect the avatar to say “thank you”. However, due to the program being automated, it is hard to program an appropriate reply at the right time. Most of the subjects do feel offended by the rudeness as they did not received a suitable feedback as they would expect in the real world.

7. Conclusion

In this chapter, we look back on what have been achieved and what have not. This is followed by some suggestions to improve this scene better for future enhancements of the project. This chapter ends with a short discussion on what the author learnt from doing this project.

7.1 An Appraisal Of The Project

Despite the difficulties encountered during these three months, this project met the initial goals. A detailed reflection was made on what have been achieved in these three months.

- **Create a scene that is similar to real life.**

A tube scene that was created for this project, the subjects had no difficulties in linking themselves to the scene. A number of subjects mentioned in the interview that the scene seemed real. However, the only flaw was the lacking of background sound which reduced the level of presence the subjects experienced.

- **Create an interesting story line that could stimulate the sense of presence for the subjects.**

Due to the fact that there is not much communication that could take place in the tube, the story line planned for the tube scene in this project was relatively un-interesting. To prevent the subjects from feeling a less sense of presence from the story line, the duration of the experiment was restricted to only one and half minutes. On the other hand, such short duration discourages the increment of the level of sense of presence.

- **Create human-like avatars with reasonable level of details.**

The avatars were created using AC3D and they are human-like. They were given facial features such as eyes, nose and mouth which reflects real humans to a degree. The joints were also implemented so that they were free to move like human would. E.g. they can rotate the shoulders.

- **Create human-like gestures for the avatars.**

The avatars in this project could breath. I.e. the torso of the avatars was expanding and contracting in a rhythmic action. The avatar, TIM, was implemented so that he rotates his head to follow subjects' movement, therefore making him more human-like.

- **Create speeches for the avatars.**

The speech was successfully created for the avatar, ZOE. The use of speech restrained the subjects moving wildly in the CAVE and engaging their attention.

- **Understand and revise the questionnaires to avoid confusion for subjects who participate in this experiment.**

The questionnaires were studied and made sure that they were fully understood by myself. However, due to the fact that the author (i.e. myself) had a more in-depth understanding of the issue, I had failed to spot the ambiguity in the questions. Almost half of the subjects who participated in the experiment experienced confusion in filling in the questionnaires.

- **Conduct the experiment in the CAVE in the middle of August.**

The experiment was carried out on the 20th and the 22nd of August. 10 subjects were recruited and completed the experiment.

- **Analyse the results from the experiment and given conclusion to the hypothesis mentioned in section 1.1.**

The completed analysis of the results are shown in the appendix in a Microsoft Office 97 Exel spreadsheet format. The analysis of the results is given in the previous chapter.

7.2 Further Improvements

If more time was available, the present project can be further improved. Here are some suggestions of improvements below:

- The experiment could be run semi-automated. I.e. the avatars that have automated behaviours could have suitable intervention issued by the lab assistant. E.g. after ZOE asked for the direction to China Town, the lab assistant could press the keyboard for an appropriated reply at suitable time.
- More facial joints could be added to the avatars to make more facial expressions, especially the movement of the mouth. It helps the subject to identify which avatar is talking to them.
- The background sound of a moving tube should be added to help the subjects to increase their level of presence. To solve the problem of the SGI machine that could only execute one sound file, maybe another workstation could be used to play the background sound. Or we could simply use a separate stereo system to play the recorded sound at the back of the CAVE.
- More joints could be added to make the avatar more human-like. E.g. the movement of hands and fingers.
- More personality can be programmed for the avatars. E.g. the male avatar that was reading on the tube could lift up his head and look at the subject when the subject was looking at him.
- More human-like gestures can be introduced such as blinking, talking to other avatars and the movement of the eyes.
- The detail of the tube train can be further refined. E.g. the separation between the seats could be added and maybe give the ability to the subject to sit on a seat on the tube train.

7.3 Experience Gained

By undertaking this project, I have learnt something important. Firstly, I learnt to schedule my time. Before working this project, I always do things at the last minute. During this three months, I learnt to spread out my task evenly and carry out the task according to the plan. Secondly, I learnt to be flexible with schedules. When the server was down in the department, I made use of the time to do other things such as read more background-related material, think and write more pseudo code for implementation and refine the design of the tube train. Thirdly, I obtained some new knowledge from doing this project. I know why and how the psychologists treat social phobia patients.

Fourthly, I learnt to implement code in C++ and how the global parameters were passed between the .cpp file to the .h header file, thus developing my programming skills at a higher level. Fifthly, I learnt to plan story lines to encourage the sense of presence and understood how the effects of presence can have an impact on peoples reactions and their interpretations. Sixthly, I enjoyed and learnt to work with others, thus further develop me ability to work with others in a team. In the first month, Laura James constructed the initial hierarchy and its class functions. I enjoyed the teamwork and learnt to programme in C++ at the same time. Finally, I learnt to run an experiment using sophisticated VR equipment and to deal with subjects. It is one of the most valuable experiences obtained from doing this project. Most of the projects for computer science do not involve other people, except the students themselves. This project provided a chance to interact with other people that were not involved in the project.

8. Appendix A- Code for cubesApp.h

```
/***************************************************************************
       Name:          Laura K James & Chien-Yu Lin
       Course:        MSc VIVE
       Title:         MSc VIVE Project: Social Phobias
       Supervisor:    Mel Slater
       Deadline:      08-09-2002
***************************************************************************/

/*************** <auto-copyright.pl BEGIN do not edit this line> **************
 *
 * VR Juggler is (C) Copyright 1998, 1999, 2000 by Iowa State University
 *
 * Original Authors:
 *   Allen Bierbaum, Christopher Just,
 *   Patrick Hartling, Kevin Meinert,
 *   Carolina Cruz-Neira, Albert Baker
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the
 * Free Software Foundation, Inc., 59 Temple Place - Suite 330,
 * Boston, MA 02111-1307, USA.
 *
 * -----------------------------------------------------------------
 * File:          cubesApp.h,v
 * Date modified: 2001/03/08 21:08:04
 * Version:       1.29.6.7
 * -----------------------------------------------------------------
 *
 *************** <auto-copyright.pl END do not edit this line> ***************/

#ifndef _CUBES_APP_
#define _CUBES_APP_

#include <vjConfig.h>

#include <GL/gl.h>
#include <GL/glu.h>
#include <vector>
#include <list>                                           // list template class
#include <sys/time.h>
#include <string>
#include <iostream>

#include <Kernel/GL/vjGlApp.h>
#include <Kernel/GL/vjGlContextData.h>
#include <Math/vjMatrix.h>
#include <Math/vjVec3.h>
#include <Kernel/vjDebug.h>

#include <Input/InputManager/vjPosInterface.h>
#include <Input/InputManager/vjAnalogInterface.h>
#include <Input/InputManager/vjDigitalInterface.h>

#include <Kernel/vjUser.h>

using namespace std;

/* 3D Vertex class */
class Vertex
{
   public:
        float x;                      // x coordinate
        float y;                      // y coordinate
        float z;                      // z coordinate

        // Set coordinates
        void input(float a, float b, float c)
        {
               x = a;
               y = b;
               z = c;
        }
};

/* RGB color class */
class Color
{
   public:
        float red;                    // Red colour
        float green;                  // Green colour
        float blue;                   // Blue colour
        float alpha;

        // Set colours
        void input(float a, float b, float c, float d)
        {
               red = a;
               green = b;
               blue = c;
               alpha = d;

        }
        void input(float a, float b, float c)
        {
               red = a;
               green = b;
               blue = c;
               alpha=1.0;
        }


};

/* Triangle class which stored 3 vertices and a colour of the triangle */
class Triangle
{
   public:
        Vertex a;                             // 1st vertex
        Vertex b;                             // 2nd vertex
        Vertex c;                             // 3rd vertex
        Color  colour;                        // colour

        // Set coordinate of vertices and the corresponding colour
        void input(Vertex d, Vertex e, Vertex f, Color h)
        {
               // Uses the input function in Vertex and Color class
               a.input(d.x, d.y, d.z);
               b.input(e.x, e.y, e.z);
               c.input(f.x, f.y, f.z);
               colour.input(h.red, h.green, h.blue,h.alpha);
        }

        // Function which draw the triangle
        void draw()
        {
               // Find the normal of the triangle for lighting
               float v1[3], v2[3], vn[3], length;

               // 1st vector
               v1[0] = a.x - b.x;
               v1[1] = a.y - b.y;
               v1[2] = a.z - b.z;

               // 2nd vector
               v2[0] = a.x - c.x;
               v2[1] = a.y - c.y;
               v2[2] = a.z - c.z;
```

```
                // Cross Product
                vn[0] = v1[1]*v2[2] - v2[1]*v1[2];
                vn[1] = v1[2]*v2[0] - v2[2]*v1[0];
                vn[2] = v1[0]*v2[1] - v2[0]*v1[1];

                // Find the length and then normalise
                length = sqrt((vn[0]*vn[0])+(vn[1]*vn[1])+(vn[2]*vn[2]));

                // Normalisation
                for(int i=0; i<3 ; i++)
                {
                        if(length != 0)
                                vn[i] /= length;
                }

                // Draw triangle
                glBegin(GL_TRIANGLES);
                        glColor4f(colour.red,colour.green,colour.blue,colour.alpha);
                // current colour
                        glNormal3fv(vn);
                        // set normal
                        glVertex3f(a.x,a.y,a.z);
                        glVertex3f(b.x,b.y,b.z);
                        glVertex3f(c.x,c.y,c.z);
                glEnd();
        }
                //scale up or down the size of the triangle
                void scale(float x1,float y1,float z1)

                {
                        a.x=a.x*x1;
                        a.y=a.y*y1;
                        a.z=a.z*z1;
                        b.x=b.x*x1;
                        b.y=b.y*y1;
                        b.z=b.z*z1;
                        c.x=c.x*x1;
                        c.y=c.y*y1;
                        c.z=c.z*z1;
                }
};


// the segment class

class Segment{

        public:
                list<Triangle> faces;         //the list of triangle


        //to sclae the size of the segment
        void scaleSeg(float x1, float y1, float z1)

        {
                list<Triangle>::iterator ptr;
                ptr = faces.begin();
                while(ptr!=faces.end())
                {       ptr->scale(x1,y1,z1);
                        ptr++;                  }
        }
};

//      Class for the joints of the avatar
//      includes class methods to rotate and translate the joints' position matrix

class Joint{

        public:

                list<Triangle> faces;         //the list of triangles
                float xRot, yRot, zRot, xPos, yPos, zPos, pxPos, pyPos, pzPos;
                                              //the original position of the joint
                                              //the rotation made and the
                                              //change in postion that needs to pass
                                              //down to the children
```

```
		list<Joint*> children;		//the pointer point to the children joint
		Segment* segment;		//the attached segment


		// rotate the joint
		void rotateJoint(float x, float y, float z){

			vjMatrix transform, tran, invtran, rotation;

			rotation.makeXYZEuler(x,y,z);

			xRot += x;
			yRot += y;
			zRot += z;

			tran.makeIdent();
			tran.setTrans(xPos,yPos,zPos);
			invtran.makeIdent();
			invtran.setTrans(-xPos,-yPos,-zPos);

			invtran.preMult(rotation);
			invtran.preMult(tran);

		//iterate the rotation to its children
			list<Joint*>::iterator c = children.begin();
			int ch = 1;
			while(c != children.end())
			{

				(*c)->rotateChild(invtran,x,y,z);

				c++;
				ch++;
			}
		}

		//rotate the children joints
		void rotateChild(vjMatrix transform, float x, float y, float z){

			float xTemp, yTemp, zTemp;

			xTemp = transform[0][0]*xPos + transform[1][0]*yPos
+transform[2][0]*zPos + transform[3][0];
			yTemp = transform[0][1]*xPos + transform[1][1]*yPos
+transform[2][1]*zPos + transform[3][1];
			zTemp = transform[0][2]*xPos + transform[1][2]*yPos
+transform[2][2]*zPos + transform[3][2];
			xPos = xTemp;
			yPos = yTemp;
			zPos = zTemp;

			xRot += x;
			yRot += y;
			zRot += z;

			list<Joint*>::iterator c = children.begin();
			while(c != children.end())
			{
				(*c)->rotateChild(transform,x,y,z);
				c++;
			}
		}

		void translateJoint(float x, float y, float z){

			xPos -= x;
			yPos -= y;
			zPos -= z;

			list<Joint*>::iterator c = children.begin();
			while(c != children.end())
			{
				(*c)->translateJoint(x,y,z);
				c++;
			}
		}
};
```

```
// the class avatar

class Avatar {

        public:
        Segment pelvis, torso, neck, skull, l_collar, l_upperarm, l_lowerarm, r_collar,
r_upperarm, r_lowerarm, l_thigh, l_calf, r_thigh, r_calf,l_feet,r_feet,l_hand,r_hand;

        Joint root, sacroilliac, l_hip, l_knee, r_hip, r_knee, vt1, skullbase,
l_sterno, l_shoulder, l_elbow, r_sterno, r_shoulder,
r_elbow,l_ankle,r_ankle,l_wrist,r_wrist;

        int currentAnim,id,test;
        unsigned long initTime, startAnimTime, currAnimTime, breath_frequency;
        Vertex neckPosition, headTracker;
        bool facing;
        float xrot, yrot,zrot;

        public:

// to create an avatar
        void create(char* sFile, char* jFile, int e_id, int frequency, Vertex neckPos,
bool facing_way){
            string bodypart;
            const char* bp;
            float xpos, ypos, zpos;         //the position of the joint of the avatar
            id=e_id;                        //the initial action id
            breath_frequency=frequency;     //how fast the avatar breath
            neckPosition=neckPos;           //the neck position
            facing=facing_way;              //the way avatar is facing
            int count = 0;                  //the time variables
            currentAnim= 0;
            startAnimTime=0;
            currAnimTime=0;
            xrot=90;                        //the rotation of the neck
            yrot=0;
            zrot=0;
            test=0;

            timeval tm;                     //to get the system time
            gettimeofday( &tm );

            initTime = tm.tv_sec;


            //initialise an array of pointers
            Segment* segArr[] = {&skull,&neck,&l_hand,&l_lowerarm, &l_upperarm,
&l_collar, &r_hand,&r_lowerarm, &r_upperarm, &r_collar, &l_feet,&l_calf,&l_thigh,
&r_feet,&r_calf, &r_thigh,&torso, &pelvis };

            ifstream segFile, jointFile;
            segFile.open(sFile);            //open the segment file

            if(!segFile)
            {
               cout<<"Cannot open file!!!"<<endl;
            }

            do
            {                               //reading the segement one by oneinto the
                    segFile >> bodypart;    //array
                    bodypart.length();
                    bp = bodypart.c_str();
                    read_segment(bp, segArr[count]);
                    count++;

             }      while((!segFile.eof()) && (count<18));

               jointFile.open(jFile);       //open the joint file
               count=0;

               if(!jointFile)
               {
                    cout<<"Cannot open Joint file!!!"<<endl;
               }
```

```
else{   //read in the joint and its position and attach the segment to
        //it, if there is a children joint, push it to the child list
        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &skullbase, &skull, xpos,ypos,zpos);


        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &vt1, &neck, xpos,ypos,zpos);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &l_wrist, &l_hand, xpos,ypos,zpos);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &l_elbow, &l_lowerarm, xpos,ypos,zpos);
        l_elbow.children.push_back(&l_wrist);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &l_shoulder, &l_upperarm, xpos,ypos,zpos);
        l_shoulder.children.push_back(&l_elbow);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &l_sterno, &l_collar, xpos,ypos,zpos);
        l_sterno.children.push_back(&l_shoulder);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &r_wrist, &r_hand, xpos,ypos,zpos);
        r_elbow.children.push_back(&r_wrist);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &r_elbow, &r_lowerarm, xpos,ypos,zpos);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &r_shoulder, &r_upperarm, xpos,ypos,zpos);
        r_shoulder.children.push_back(&r_elbow);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &r_sterno, &r_collar, xpos,ypos,zpos);
        r_sterno.children.push_back(&r_shoulder);

        vt1.children.push_back(&skullbase);
        vt1.children.push_back(&l_sterno);
        vt1.children.push_back(&r_sterno);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &l_ankle, &l_feet, xpos,ypos,zpos);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
        read_joint(bp, &l_knee, &l_calf, xpos,ypos,zpos);
        l_knee.children.push_back(&l_ankle);

        jointFile>>bodypart>>xpos>>ypos>>zpos;
        bodypart.length();
        bp = bodypart.c_str();
```

```
                read_joint(bp, &l_hip, &l_thigh, xpos,ypos,zpos);
                l_hip.children.push_back(&l_knee);

                jointFile>>bodypart>>xpos>>ypos>>zpos;
                bodypart.length();
                bp = bodypart.c_str();
                read_joint(bp, &r_ankle, &r_feet, xpos,ypos,zpos);
                jointFile>>bodypart>>xpos>>ypos>>zpos;
                bodypart.length();
                bp = bodypart.c_str();
                read_joint(bp, &r_knee, &r_calf, xpos,ypos,zpos);
                r_knee.children.push_back(&r_ankle);

                jointFile>>bodypart>>xpos>>ypos>>zpos;
                bodypart.length();
                bp = bodypart.c_str();
                read_joint(bp, &r_hip, &r_thigh, xpos,ypos,zpos);
                r_hip.children.push_back(&r_knee);

                jointFile>>bodypart>>xpos>>ypos>>zpos;
                bodypart.length();
                bp = bodypart.c_str();
                read_joint(bp, &root, &torso, xpos,ypos,zpos);
                root.children.push_back(&vt1);

                jointFile>>bodypart>>xpos>>ypos>>zpos;
                bodypart.length();
                bp = bodypart.c_str();
                read_joint(bp, &sacroilliac, &pelvis, xpos,ypos,zpos);

                sacroilliac.children.push_back(&root);
                sacroilliac.children.push_back(&l_hip);
                sacroilliac.children.push_back(&r_hip);

                jointFile.close();
        }

}
/* Read a 3D model from files and push triangle in the corresponding list */
void read_segment(const char* file, Segment* seg){

    float c1[3], c2[3], c3[3];              // arrays which read in the
                                            //coordinates of vertices
    string rgb;                             // string which read in the colour
                                            //string
    Vertex ver_1, ver_2, ver_3;             // 3 vertices
    Color tri_color;                        // colour of the triangle
    Triangle tri;                           // create a triangle

    ifstream inFile;
    inFile.open(file);                      // Open file

    if(!inFile)
    {
                cout<<"Cannot open Segment file!!!"<<endl;
    }

        // Read each triangle from file and push it in the list
    do
    {

                inFile>>c1[0]>>c1[1]>>c1[2];          // read vertex one
                inFile>>c2[0]>>c2[1]>>c2[2];          // read vertex two
                inFile>>c3[0]>>c3[1]>>c3[2];          // read vertex three
                inFile>>rgb;                          // read colour


                // Change the colour string which to r,g,b colours
                float temp1 = hexdec((char)rgb[2], (char)rgb[3]);
                float temp2 = hexdec((char)rgb[4], (char)rgb[5]);
                float temp3 = hexdec((char)rgb[6], (char)rgb[7]);

                // Put coordinates in each vertices
                ver_1.input(c1[0],c1[1],c1[2]);
                ver_2.input(c2[0],c2[1],c2[2]);
                ver_3.input(c3[0],c3[1],c3[2]);
```

```
                    // Set the colour of the triangle
                    tri_color.input(temp1, temp2, temp3);

                    // Input 3 vertices and a colour to the triangle
                    tri.input(ver_1, ver_2, ver_3, tri_color);

                    // Push the triangle to the back of the list
                    seg->faces.push_back(tri);

           }while(!inFile.eof());

       }

       /* Read a 3D model from files and push triangle in the corresponding list */
       void read_joint(const char* file, Joint* aJoint, Segment* seg, float xTrans,
float yTrans, float zTrans)
       {
               float c1[3], c2[3], c3[3];    // arrays which read in the coordinates
of vertices
           string rgb;                                      // string which read in
the colour string
               Vertex ver_1, ver_2, ver_3;   // 3 vertices
               Color tri_color;              // colour of the triangle
               Triangle tri;                 // create a triangle
               aJoint->xRot = 0;
               aJoint->yRot = 0;
               aJoint->zRot = 0;
               aJoint->xPos = xTrans;
               aJoint->yPos = yTrans;
               aJoint->zPos = zTrans;
               aJoint->segment = seg;

           ifstream inFile;
           inFile.open(file);                      // Open file

           if(!inFile)
           {
                    cout<<"Cannot open Joint file!!!"<<endl;
           }

               // Read each triangle from file and push it in the list
           do
           {
                    inFile>>c1[0]>>c1[1]>>c1[2];  // read vertex one
                    inFile>>c2[0]>>c2[1]>>c2[2];  // read vertex two
                    inFile>>c3[0]>>c3[1]>>c3[2];  // read vertex three
                    inFile>>rgb;                  // read colour

                    // Change the colour string which to r,g,b colours
                    float temp1 = hexdec((char)rgb[2], (char)rgb[3]);
                    float temp2 = hexdec((char)rgb[4], (char)rgb[5]);
                    float temp3 = hexdec((char)rgb[6], (char)rgb[7]);

                    // Put coordinates in each vertices
                    ver_1.input(c1[0],c1[1],c1[2]);
                    ver_2.input(c2[0],c2[1],c2[2]);
                    ver_3.input(c3[0],c3[1],c3[2]);

                    // Set the colour of the triangle
                    tri_color.input(temp1, temp2, temp3);

                    // Input 3 vertices and a colour to the triangle
                    tri.input(ver_1, ver_2, ver_3, tri_color);

                    // Push the triangle to the back of the list
                    aJoint->faces.push_back(tri);

           }while(!inFile.eof());

       }

       // to make the personality of the avatar

       void actions(int actionID){
               if(startAnimTime==0){

                    timeval tm;
```

```
                gettimeofday( &tm );

                unsigned long milli = tm.tv_usec;
                unsigned long sec1 = tm.tv_sec;
                unsigned long diff = sec1-initTime;

                startAnimTime = (diff*1000000)+milli;

        }
        timeval tm;
        gettimeofday( &tm );

        /*unsigned long milli = tm.tv_usec;
        unsigned long sec2 = tm.tv_sec;
        unsigned long diff = sec2-initTime;

        unsigned long now = (diff*1000000)+milli;
        currAnimTime = now-startAnimTime;*/

        switch(actionID){
                case 0:
                                break;
                case 1: breath();
                                break;
                case 2: sleeping();
                                break;
                case 3: looking();
                                break;
                case 4: ask();
                                break;
                case 5: hey();
                                break;

        }
}

void breath()
{               long currAnimTime=getTime();
                int ans;
                ans=currAnimTime/breath_frequency;
                if ((ans%2)==0)
                {torso.scaleSeg(1.003,1,1.003);}
                else
                {torso.scaleSeg((1/1.003),1,(1/1.003));}}

//to make ZOE ask question when the subject try to pass TIM
void ask(){
                float xrot1,yrot1,xrot2,yrot2;
                Vertex temp;
                const float pi=atan(1.0)*4;
                temp=unit_vec(neckPosition,headTracker);


        //to make ZOE look at the subject when the subject gets near to ZOE
                if ((headTracker.x<1.2)&&(abs(headTracker.z)<0.08))
                {

                        yrot1=sin(temp.y);  //<---- need to convert to degrees;

                        yrot2=(yrot1/pi)*180;

                        if(abs(yrot2-yrot)>0.00001)
                        {       skullbase.rotateJoint(yrot2-yrot,0,0);
                                yrot=yrot2;                     }

                        xrot1=acos(temp.x/cos(yrot1));
                        xrot2=(xrot1/pi)*180;
                        if (xrot2<0)
                        {xrot2=180+xrot2;}
                        if(abs(xrot2-xrot)>0.00001)
                        {
                                skullbase.rotateJoint(0,-1*(xrot-xrot2),0);
                                xrot=xrot2;}
                        if((test==0)&&(headTracker.x<0.2))
                        {       r_shoulder.rotateJoint(-10,0,0);
                        system("soundplayer -nodisplay ask.wav&");
                                test++;}
```

```
			cout<< "OVER THE LIMIT!\n";
			}
		else
		{
		if(test!=0)
		{	r_shoulder.rotateJoint(10,0,0);
			test--;}
		if (xrot!=90)
		{	skullbase.rotateJoint(0,90-xrot,0);
			xrot=90;	}
		if (yrot!=0)
		{	skullbase.rotateJoint(-yrot,0,0);
			yrot=0;		}}
		long currAnimTime=getTime();
		int ans;
		ans=currAnimTime/breath_frequency;
		if ((ans%2)==0)
		{torso.scaleSeg(1.003,1,1.003);}
		else
		{torso.scaleSeg((1/1.003),1,(1/1.003));}
		currentAnim=0;
			}
//to make BOB nodding to sleep
void sleeping(){
		long currAnimTime=getTime();
		int ans1,ans2;
		ans1=currAnimTime/200000;
		if ((ans1%2)==0)
		skullbase.rotateJoint(2,0,0);
		else
		skullbase.rotateJoint(-2,0,0);

		ans2=currAnimTime/breath_frequency;
		if ((ans2%2)==0)
		{torso.scaleSeg(1.003,1,1.003);}
		else
		{torso.scaleSeg((1/1.003),1,(1/1.003));}


}
//to make TIM look at the subject
void looking(){
		long currAnimTime=getTime();
		int ans1;
		float xrot1,yrot1,xrot2,yrot2;
		Vertex temp;
		const float pi=atan(1.0)*4;
		temp=unit_vec(neckPosition,headTracker);

		yrot1=sin(temp.y);  //<---- need to convert to degrees;

		yrot2=(yrot1/pi)*180;

		if(abs(yrot2-yrot)>0.0000000001)
		{	skullbase.rotateJoint(yrot2-yrot,0,0);
			yrot=yrot2;			}

		xrot1=acos(temp.x/cos(yrot1));
		xrot2=(xrot1/pi)*180;
		if (xrot2<0)
		{xrot2=180+xrot2;}
		if(abs(xrot2-xrot)>0.0000000001)
		{
			skullbase.rotateJoint(0,-1*(xrot-xrot2),0);
			xrot=xrot2;			}

		ans1=currAnimTime/breath_frequency;
		if ((ans1%2)==0)
		{torso.scaleSeg(1.003,1,1.003);}
		else
		{torso.scaleSeg((1/1.003),1,(1/1.003));}


}

Vertex unit_vec(Vertex v1, Vertex v2)
```

```
        {       float sum;
                Vertex v3;
                v3=subtract(v1,v2);
                sum=vec_length(v3);
                v3.x=v3.x/sum;
                v3.y=v3.y/sum;
                v3.z=v3.z/sum;
                return v3;
        }

        float vec_length(Vertex v3)
        {       float sum;
                sum=v3.x*v3.x+v3.y*v3.y+v3.z*v3.z;
                return sum;
        }

        Vertex subtract(Vertex v1, Vertex v2)

        {
                Vertex v3;
                v3.x=v1.x-v2.x;
                v3.y=v1.y-v2.y;
                v3.z=v1.z-v2.z;
                return v3;
        }
        long getTime()

        {               timeval tm;
                        gettimeofday( &tm );

                        unsigned long milli = tm.tv_usec;
                        unsigned long sec2 = tm.tv_sec;
                        unsigned long diff = sec2-initTime;

                        unsigned long now = (diff*1000000)+milli;
                        unsigned long currAnimTime = now-startAnimTime;
                        return currAnimTime;
        }

        // Function which change hex. colours to normal r,g,b representation
        float hexdec(char a, char b)
        {
            float dec,c,d;

            if(a>'9')
               c = a - 'A' + 10;
            else
               c = a - '0';

            if(b>'9')
               d = b - 'A' + 10;
            else
               d = b - '0';

            dec = (c*16 + d)/256;

            return dec;
        }
};

// Class to hold all context specific data
class ContextData
{
public:
   ContextData()
   {
      dlIndex  = -1;
      maxIndex = -1;
   }

public:
   int   dlIndex;
   int   maxIndex;     // For debugging purposes only!
};

// Class to hold all data for a specific user
class UserData
```

```
{
public:
  UserData(vjUser* user, std::string wandName, std::string incButton,
           std::string decButton, std::string stopButton,
           std::string joystickX, std::string joystickY );

  // Update the navigation matrix
  //
  // Uses a quaternion to do rotation in the environment
  void updateNavigation();

public:
      // Devices to use for the given user
   vjPosInterface       mWand;                  // the Wand
   vjDigitalInterface   mIncVelocityButton;     // Button for velocity
   vjDigitalInterface   mDecVelocityButton;
   vjDigitalInterface   mStopButton;            // Button to stop
   vjAnalogInterface    mJoystick[2];           // Joystick

      // Navigation info for the user
   float                mCurVelocity;  // The current velocity
   vjMatrix             mNavMatrix;    // Matrix for navigation in the application
   vjUser*              mUser;         // The user we hold data for

   // Defaults for joystick navigation
   float mSpeedMax;
   float mSpeedExponent;
   float mRotateSpeedMax;
   float mRotateSpeedExponent;

};

//----------------------------------------------------
// Demonstration OpenGL application class
//
// This application simply renders a field of cubes.
//-----------------------------------------------------
class cubesApp : public vjGlApp
{

public:
   cubesApp(vjKernel* kern) : vjGlApp(kern)
   {
      ;
   }

   virtual ~cubesApp() {}

   // Execute any initialization needed before the API is started.  Put device
   // initialization here.
   virtual void init();

   // Execute any initialization needed <b>after</b> API is started
   //  but before the drawManager starts the drawing loops.
   virtual void apiInit()
   {
      vjDEBUG(vjDBG_ALL,0) << "---- cubesApp::apiInit() ----\n" << vjDEBUG_FLUSH;
   }

   // Called immediately upon opening a new OpenGL context.  This is called
   // once for every display window that is opened.  Put OpenGL resource
   // allocation here.
   virtual void contextInit();

   // Called immediately upon closing an OpenGL context
   // (called for every window that is closed)
   // put your opengl deallocation here...
   virtual void contextClose();

   /**   name Drawing Loop Functions
    *
    *  The drawing loop will look similar to this:
    *
    *  while (drawing)
    *  {
    *        preFrame();
    *       draw();
    *        intraFrame();      // Drawing is happening while here
    *       sync();
    *        postFrame();      // Drawing is now done
    *
    *       UpdateTrackers();
    *  }
    *
    */

   // Function called after tracker update but before start of drawing.  Do
   // calculations and state modifications here.
   virtual void preFrame()
   {
       vjDEBUG(vjDBG_ALL,5) << "cubesApp::preFrame()" << std::endl
                           << vjDEBUG_FLUSH;

       for(unsigned int i=0;i<mUserData.size();i++)
        {mUserData[i]->updateNavigation();       // Update the navigation matrix
        //globalWand = mUserData[i]->mWand->getData();
       globalWand = mUserData[i]->mUser->getHeadPos();
                                                             }

               Vertex wVer;
               wVer.input(globalWand->mat[3][0],globalWand->mat[3][1],globalWand-
>mat[3][2]);
cout <<"wand position:"<<globalWand->mat[3][0]<<" "<<globalWand->mat[3][1]<<"
"<<globalWand->mat[3][2];
               list<Avatar*>::iterator person =  People.begin();

               Sue.currentAnim = 1;   //SUE is breathing
               Tim.currentAnim = 3;   //TIM is looked at the subject
               Bob.currentAnim = 2;   //BOB is sleeping
               Zoe.currentAnim = 4;   //ZOE ask the question

               while(person != People.end()){
                       (*person)->headTracker=wVer;
                       (*person)->actions((*person)->currentAnim);
                       person++;
                       }
               //when the duration of the experiment is 60 seconds
               if (((Bob.getTime())>60000000)&&(count<11))
               {
                       if (count==1)
                       //play the message to inform the subject the end of experiment
                       {       system("soundplayer -nodisplay end.wav&");}

                       count++;}

   }

   // Function to draw the scene.  Put OpenGL draw functions here.
   //
   // PRE: OpenGL state has correct transformation and buffer selected
   // POST: The current scene has been drawn
   virtual void draw()
   {
      initGLState();    // This should really be in another function

      myDraw(vjGlDrawManager::instance()->currentUserData()->getUser());
   }

   /// Function called after drawing has been triggered but BEFORE it completes
   virtual void intraFrame()
   {
      vjDEBUG(vjDBG_ALL,5) << "cubesApp::intraFrame()" << std::endl
                          << vjDEBUG_FLUSH;
   }

   // Function called before updating trackers but after the frame is drawn.
   // Do calculations here.
   virtual void postFrame()
   {

      vjDEBUG(vjDBG_ALL,5) << "cubesApp::postFrame" << std::endl
                          << vjDEBUG_FLUSH;

        }
   //: Make sure that all our dependencies are satisfied
   // Make sure that there are vjUsers registered with the system
   virtual bool depSatisfied()
   {
      // We can't start until there are users registered wth the system
      // We rely upon users to keep track of the multi-user data structure
      int num_users = vjKernel::instance()->getUsers().size();
      return (num_users > 0);
   }

//   virtual void read_segment(char* file, Segment* seg);
//   virtual void read_joint(char* file, Joint* aJoint, Segment* seg, float xTrans,
float yTrans);

 virtual void read_file(char* file, list<Triangle>& tl);
 virtual float vec_length(Vertex v3);
 virtual Vertex unit_vec(Vertex v1, Vertex v2);
 virtual Vertex subtract(Vertex v1, Vertex v2);

private:

   //----------------------------------------------
   //  Draw the scene.  A bunch of boxes of differing color and stuff
   //----------------------------------------------
   void myDraw(vjUser* user);
   void initGLState();

   void drawCube()
   {

     //drawbox(-1.0f, 1.0f, -1.0f, 1.0f, -1.0f, 1.0f, GL_QUADS);
   }

   void drawbox(GLdouble x0, GLdouble x1, GLdouble y0, GLdouble y1,
                GLdouble z0, GLdouble z1, GLenum type);

public:
   vjGlContextData<ContextData>  mDlData;       // Data for display lists
   vjGlContextData<ContextData>  mDlDebugData;  // Data for debugging display lists
   std::vector<UserData*>        mUserData;     // All the users in the program

   list<Avatar*>                            People;
   vjMatrix*                                globalWand;
   list<Triangle>                           tube;
   list<Triangle>                           tube_door;
   Vertex                                   wVer;
   Avatar                                   Bob;
   Avatar                                   Sue;
   Avatar                                   Tim;
   Avatar                                   Zoe;
   int                                      count;

};


#endif
```

9. Appendix B- Code For cubesApp.cpp

```
/*************************************************************************
***
	Name:		Laura James, Chien-Yu Lin
	Course:		MSc VIVE
	Title:		MSc project: Social Phobia
	superviser:	Mel Slater
	Deadline:	07-09-2002
*************************************************************************
**/

/*************** <auto-copyright.pl BEGIN do not edit this line>
**************
 *
 * VR Juggler is (C) Copyright 1998, 1999, 2000 by Iowa State University
 *
 * Original Authors:
 *   Allen Bierbaum, Christopher Just,
 *   Patrick Hartling, Kevin Meinert,
 *   Carolina Cruz-Neira, Albert Baker
 *
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the
 * Free Software Foundation, Inc., 59 Temple Place - Suite 330,
 * Boston, MA 02111-1307, USA.
 *
 * -----------------------------------------------------------------
 * File:          cubesApp.cpp,v
 * Date modified: 2001/04/06 21:45:18
 * Version:       1.6.6.6
 * -----------------------------------------------------------------
 *
 *************** <auto-copyright.pl END do not edit this line>
***************/

#include <iostream>
#include <math.h>
#include <fstream>
#include <string>
#include <list>
#include <Math/vjQuat.h>
#include <cubesApp.h>
using namespace std;

/* function declarations */
void multMatrix(float ans[], vjMatrix* mat1, float* mat2);
void read_file(char* file, list<Triangle>& tl);
float vec_length(Vertex v3);
Vertex unit_vec(Vertex v1, Vertex v2);
Vertex subtract(Vertex v1, Vertex v2);
float hexdec(char a, char b);

// -----------------------------------------------------------------------
---
```

```
// UserData methods.
// ----------------------------------------------------------------------------
---

// Constructor
// Takes the string names of the devices to use
// NOTE: This means that we cannot construct a user until the input manager
is loaded
// Ex. The Init function

UserData::UserData(vjUser* user, std::string wandName, std::string
incButton,
        std::string decButton, std::string stopButton,
        std::string joystickX, std::string joystickY )
{
  mNavMatrix.makeIdent();
  mUser = user;

  // Initialize devices
  mWand.init(wandName);
  mIncVelocityButton.init(incButton);
  mDecVelocityButton.init(decButton);
  mStopButton.init(stopButton);
  mJoystick[0].init(joystickX);
  mJoystick[1].init(joystickY);

  // Defaults for joystick navigation
  mSpeedMax=2.0;
  mSpeedExponent=2.0;
  mRotateSpeedMax=5.0;
  mRotateSpeedExponent=2.0;
}


// Update the navigation matrix
// Uses a quaternion to do rotation in the environment
void UserData::updateNavigation()
{
  vjMatrix    local_xform;
  // --------------------------------------------------------------------------
  // joystick-based navigation...
  float x, y;
  x = mJoystick[0]->getData();
  y = mJoystick[1]->getData();

   if (x<0.45)
     x += 0.05;
   else if (x<0.55)
     x = 0.5;
   else
     x -= 0.05;
   if (y<0.45)
     y += 0.05;
   else if (y<0.55)
     y = 0.5;
   else
       y -= 0.05;

   vjMatrix* wandPosInt = mWand->getData();
   vjMatrix* headPosInt = mUser->getHeadPos();

   /* transform in cave local coordinates,
      i.e. rotate about centre of reactor, and translate in direction
      of centre screen */

   float rotateSpeed= mRotateSpeedMax*(x-0.5);
   if (mRotateSpeedExponent!=1.0) {
     if (rotateSpeed>=0)
```

```
        rotateSpeed = pow(rotateSpeed, mRotateSpeedExponent);
      else
        rotateSpeed = -pow(-rotateSpeed, mRotateSpeedExponent);
    }

    local_xform.makeXYZEuler(0,rotateSpeed,0);

    local_xform.preTrans((*headPosInt).mat[3][0],
                     (*headPosInt).mat[3][1],
                     (*headPosInt).mat[3][2],
                     local_xform);
    local_xform.postTrans(local_xform,
                      -(*headPosInt).mat[3][0],
                      -(*headPosInt).mat[3][1],
                      -(*headPosInt).mat[3][2]);

    float speed = -((y-0.5)/2.0)*mSpeedMax;
    if (mSpeedExponent!=1.0) {
      if (speed>=0)
        speed = pow(speed, mSpeedExponent);
      else
        speed = -pow(-speed, mSpeedExponent);
    }

    // Always follows ground
    local_xform.postTrans(local_xform,
                      -(*headPosInt).mat[2][0]*speed,
                      0,
                      -(*headPosInt).mat[2][2]*speed);
    mNavMatrix.preMult(local_xform);


}

// -----------------------------------------------------------------------------
---
// cubesApp methods.
// -----------------------------------------------------------------------------
---

// Execute any initialization needed before the API is started.  Put device
// initialization here.
void cubesApp::init(){

// Read the 3D model from files and put triangle in each lists

       read_file("tube.tri",tube);
       read_file("tube_door.tri",tube_door);
       Vertex neck1, neck2, neck3,neck4;
// Initialise the neck position of the avatars

       neck1.x=0;
       neck1.y=1.01;
       neck1.z=-0.8;

       neck2.x=0;
       neck2.y=1.11;
       neck2.z=0.8;

       neck3.x=1;
       neck3.y=0.81;
       neck3.z=-0.03;

       neck4.x=0.33;
       neck4.y=1.35;
       neck4.z=-0.68;

// create the avatars by calling the Avatar Class member function: create
// create (“the segment file”,”the joint file”,the way the avatar is
facing,the breathing frquency, the neck //              postiotns);

       Tim.create("tim1.dat", "tim2.dat",1,2500000, neck1,true);   //create
TIM
       People.push_back(&Tim);
       Sue.create("sue1.dat", "sue2.dat",1,2800000, neck2,false);  //create
SUE
       People.push_back(&Sue);
       Bob.create("bob1.dat", "bob2.dat",1,3000000, neck3,true);   //create
BOB
       People.push_back(&Bob);
       Zoe.create("zoe1.dat", "zoe2.dat",1,2700000, neck4,true);   //create
ZOE
       People.push_back(&Zoe);

// initialise the default posture of the avatars
//always rotate the children joints before rotate the parent joints

       Bob.sacroilliac.translateJoint(0,Bob.l_knee.yPos-0.055,0);  //BOB is
sitting
       Bob.l_elbow.rotateJoint(-80,0,0);
       Bob.r_elbow.rotateJoint(-80,0,0);
       Bob.l_elbow.rotateJoint(0,10,0);
       Bob.r_elbow.rotateJoint(0,-10,0);
       Bob.l_shoulder.rotateJoint(-30,0,0);
       Bob.r_shoulder.rotateJoint(-30,0,0);
       Bob.l_knee.rotateJoint(90,0,0);
       Bob.r_knee.rotateJoint(90,0,0);
       Bob.l_hip.rotateJoint(-90,0,0);
       Bob.r_hip.rotateJoint(-90,0,0);

       Tim.sacroilliac.translateJoint(0,Tim.l_knee.yPos-0.055,0);  //TIM is
sitting
       Tim.l_elbow.rotateJoint(-80,0,0);
       Tim.r_elbow.rotateJoint(-80,0,0);
       Tim.l_elbow.rotateJoint(0,10,0);
       Tim.r_elbow.rotateJoint(0,-10,0);
       Tim.l_shoulder.rotateJoint(-30,0,0);
       Tim.r_shoulder.rotateJoint(-30,0,0);
       Tim.l_knee.rotateJoint(90,0,0);
       Tim.r_knee.rotateJoint(90,0,0);
       Tim.l_hip.rotateJoint(-90,0,0);
       Tim.r_hip.rotateJoint(-90,0,0);

       Sue.sacroilliac.translateJoint(0,Sue.l_knee.yPos,0);        //SUE is
sitting
       Sue.l_shoulder.rotateJoint(30,0,0);
       Sue.r_shoulder.rotateJoint(30,0,0);
       Sue.sacroilliac.rotateJoint(0,180,0);

       count=0;

   vjDEBUG(vjDBG_ALL,0) << "---------- cubes:App:init() ---------------"
                     << std::endl << vjDEBUG_FLUSH;
   std::vector<vjUser*> users = kernel->getUsers();
   int num_users = users.size();
   vjASSERT(num_users > 0);             // Make sure that we actually have
users defined

   UserData* new_user=NULL;
   mUserData = std::vector<UserData*>(num_users);

   switch (num_users)
   {
   case (2):
      new_user = new UserData(users[1],"VJWand1", "VJButton0_1",
"VJButton1_1",
                              "VJButton2_1", "VJJoystickX_1",
"VJJoystickY_1");
      mUserData[1] = new_user;
      vjASSERT(users[1]->getId() == 1);
   case (1):
      new_user = new UserData(users[0],"VJWand", "VJButton0", "VJButton1",
                              "VJButton2", "VJJoystickX", "VJJoystickY");
      mUserData[0] = new_user;
      vjASSERT(users[0]->getId() == 0);
      break;
   default:
      vjDEBUG(vjDBG_ALL,0) << clrOutNORM(clrRED, "ERROR:") << " Bad number
of users." << vjDEBUG_FLUSH;
      exit();
      break;
   }

}


// Called immediately upon opening a new OpenGL context.  This is called
once
// for every display window that is opened.  Put OpenGL resource allocation
// here.
void cubesApp::contextInit()
{
}

//: Called immediately upon closing an OpenGL context
// (called for every window that is closed)
// put your opengl deallocation here...
void cubesApp::contextClose()
{
}

//----------------------------------------------
// Draw the scene.  A bunch of boxes of
// differing color and stuff.
//----------------------------------------------
void cubesApp::myDraw(vjUser* user)
{
   vjDEBUG(vjDBG_ALL,2) << "\n--- myDraw() ---\n" << vjDEBUG_FLUSH;

   //static const float SCALE = 100;
   //static const float SCALE = 10;
   //static const float INCR = 0.1;

   glClearColor(0.3, 0.3, 0.7, 0.0);
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   for(unsigned int i=0;i<mUserData.size();i++)
      globalWand = mUserData[i]->mWand->getData();       // Update the
navigation matrix

   glPushMatrix();

   vjMatrix nav_matrix = mUserData[user->getId()]->mNavMatrix;
  glMultMatrixf(nav_matrix.getFloatPtr());

    // Initialise a iterator(pointer) for each list
    // Each pointer points to the beginning of the list at the beginning

   list<Triangle>::iterator lwrist;            //the left wrist
   list<Triangle>::iterator rwrist;            //the right wrist
   list<Triangle>::iterator lhand;             //the left hand
   list<Triangle>::iterator rhand;             //the right hand
   list<Triangle>::iterator lshoulder;         //the left shoulder
   list<Triangle>::iterator luarm;             //the left arm
   list<Triangle>::iterator lelbow;           //the left elbow
   list<Triangle>::iterator larm;       //the left arm
   list<Triangle>::iterator lsterno;          //the left sterno
   list<Triangle>::iterator lcollar;          //the left collar
   list<Triangle>::iterator rshoulder;        //the right shoulder
   list<Triangle>::iterator ruarm;            //the right arm
   list<Triangle>::iterator relbow;           //the right elbow
   list<Triangle>::iterator rarm;       //the right arm
   list<Triangle>::iterator rsterno;          //the right sterno
   list<Triangle>::iterator rcollar;          //the right collar
   list<Triangle>::iterator pvt1;       //the vt1
   list<Triangle>::iterator pneck;            //the neck
   list<Triangle>::iterator sb;         //the skull base
   list<Triangle>::iterator ps;         //the skull
   list<Triangle>::iterator proot;            //the root
   list<Triangle>::iterator ptorso;           //the torso
   list<Triangle>::iterator sacro;            //the sacrollium
   list<Triangle>::iterator pp;         //the pelvis
   list<Triangle>::iterator lhip;       //the left hip
   list<Triangle>::iterator lthigh;           //the left thigh
   list<Triangle>::iterator lknee;            //the left knee
   list<Triangle>::iterator lcalf;            //the left calf
   list<Triangle>::iterator rhip;       //the right hip
   list<Triangle>::iterator rthigh;           //the right thigh
   list<Triangle>::iterator rknee;            //the right knee
   list<Triangle>::iterator rcalf;            //the right calf
   list<Triangle>::iterator lankle;           //the left ankle
   list<Triangle>::iterator rankle;           //the right ankle
   list<Triangle>::iterator lfeet;            //the left foot
   list<Triangle>::iterator rfeet;            //the right foot
   list<Avatar*>::iterator person = People.begin();

   list<Triangle>::iterator the_tube=tube.begin();
       while (the_tube!=tube.end())
             {      glPushMatrix();
             the_tube->draw();
             the_tube++;
             glPopMatrix();
       }

    list<Triangle>::iterator door=tube_door.begin();
       while (door!=tube_door.end())
             {      glPushMatrix();
             if (count==1)
             {glTranslatef(-1*(count-1)*0.114,0,0);}        //to open the
door of the tube
             glTranslatef(-1*count*0.114,0,0);              //train after
the broadcast
             door->draw();                                  // message is
played
             door++;
             glPopMatrix();
       }

   // Assign the iterator to the beginning of the triangle list

   while(person != People.end()){

       lshoulder = (*person)->l_shoulder.faces.begin();
       luarm = (*person)->l_upperarm.faces.begin();
       lelbow = (*person)->l_elbow.faces.begin();
       larm = (*person)->l_lowerarm.faces.begin();
       lsterno = (*person)->l_sterno.faces.begin();
       lcollar = (*person)->l_collar.faces.begin();
       rshoulder = (*person)->r_shoulder.faces.begin();
       ruarm = (*person)->r_upperarm.faces.begin();
       relbow = (*person)->r_elbow.faces.begin();
       rarm = (*person)->r_lowerarm.faces.begin();
```

```
        rsterno = (*person)->r_sterno.faces.begin();
        rcollar = (*person)->r_collar.faces.begin();
        pvt1 = (*person)->vt1.faces.begin();
        pneck = (*person)->neck.faces.begin();
        sb = (*person)->skullbase.faces.begin();
        ps = (*person)->skull.faces.begin();
        proot = (*person)->root.faces.begin();
        ptorso = (*person)->torso.faces.begin();
        sacro = (*person)->sacroilliac.faces.begin();
        pp = (*person)->pelvis.faces.begin();
        lhip = (*person)->l_hip.faces.begin();
        lthigh = (*person)->l_thigh.faces.begin();
        lknee = (*person)->l_knee.faces.begin();
        lcalf = (*person)->l_calf.faces.begin();
        rhip = (*person)->r_hip.faces.begin();
        rthigh = (*person)->r_thigh.faces.begin();
        rknee = (*person)->r_knee.faces.begin();
        rcalf = (*person)->r_calf.faces.begin();
        lankle = (*person)->l_ankle.faces.begin();
        rankle = (*person)->r_ankle.faces.begin();
        lfeet = (*person)->l_feet.faces.begin();
        rfeet = (*person)->r_feet.faces.begin();
        lwrist=(*person)->l_wrist.faces.begin();
        rwrist=(*person)->r_wrist.faces.begin();
        lhand=(*person)->l_hand.faces.begin();
        rhand=(*person)->r_hand.faces.begin();

//draw the parts of the avatar
//firstly, translate the joint to its proper position
//secondly, draw the joint follow by the attaching segment

        glPushMatrix();            //draw vt1 and the neck

        glTranslatef((*person)->vt1.xPos,(*person)->vt1.yPos,(*person)->
vt1.zPos);
        glRotatef((*person)->vt1.xRot,1,0,0);
        glRotatef((*person)->vt1.yRot,0,1,0);
        glRotatef((*person)->vt1.zRot,0,0,1);


        /*while(pvt1 != (*person)->vt1.faces.end())
        {
              pvt1->draw();
              pvt1++;
        }*/
        while(pneck != (*person)->neck.faces.end())
        {
              pneck->draw();
              pneck++;
        }
        glPopMatrix();

        glPushMatrix();            //draw the skull base and the skull

        glTranslatef((*person)->skullbase.xPos,(*person)->
skullbase.yPos,(*person)->skullbase.zPos);
        glRotatef((*person)->skullbase.xRot,1,0,0);
        glRotatef((*person)->skullbase.yRot,0,1,0);
        glRotatef((*person)->skullbase.zRot,0,0,1);

        /*while(sb != (*person)->skullbase.faces.end())
        {
              sb->draw();
              sb++;
        }*/
        while(ps != (*person)->skull.faces.end())
        {
              ps->draw();
```

```
			ps++;
		}
		glPopMatrix();

		glPushMatrix();			//draw vt1 and the neck

		glTranslatef((*person)->l_sterno.xPos,(*person)->
	l_sterno.yPos,(*person)->l_sterno.zPos);
		glRotatef((*person)->l_sterno.xRot,1,0,0);
		glRotatef((*person)->l_sterno.yRot,0,1,0);
		glRotatef((*person)->l_sterno.zRot,0,0,1);

		/*while(lsterno != (*person)->l_sterno.faces.end())
		{
			lsterno->draw();
			lsterno++;
		}
		while(lcollar != (*person)->l_collar.faces.end())
		{
			lcollar->draw();
			lcollar++;
		}*/
		glPopMatrix();

		glPushMatrix();		//draw left shoulder and left upper arm

		glTranslatef((*person)->l_shoulder.xPos,(*person)->
	l_shoulder.yPos,(*person)->l_shoulder.zPos);
		glRotatef((*person)->l_shoulder.xRot,1,0,0);
		glRotatef((*person)->l_shoulder.yRot,0,1,0);
		glRotatef((*person)->l_shoulder.zRot,0,0,1);

		while(lshoulder != (*person)->l_shoulder.faces.end())
		{
			lshoulder->draw();
			lshoulder++;
		}
		while(luarm != (*person)->l_upperarm.faces.end())
		{
			luarm->draw();
			luarm++;
		}

		glPopMatrix();

		glPushMatrix();		//draw left elbow and left lower arm

		glTranslatef((*person)->l_elbow.xPos,(*person)->
	l_elbow.yPos,(*person)->l_elbow.zPos);

		glRotatef((*person)->l_elbow.xRot,1,0,0);
		glRotatef((*person)->l_elbow.yRot,0,1,0);
		glRotatef((*person)->l_elbow.zRot,0,0,1);
		while(lelbow != (*person)->l_elbow.faces.end())
		{
			lelbow->draw();
			lelbow++;
		}
		while(larm != (*person)->l_lowerarm.faces.end())
		{
			larm->draw();
			larm++;
		}

		glPopMatrix();

		glPushMatrix();		//draw left wrist and left hand
	glTranslatef((*person)->l_wrist.xPos,(*person)->
l_wrist.yPos,(*person)->l_wrist.zPos);
	glRotatef((*person)->l_wrist.xRot,1,0,0);
	glRotatef((*person)->l_wrist.yRot,0,1,0);
	glRotatef((*person)->l_wrist.zRot,0,0,1);
	while(lwrist != (*person)->l_wrist.faces.end())
	{
		lwrist->draw();
		lwrist++;
	}
	while(lhand != (*person)->l_hand.faces.end())
	{
		lhand->draw();
		lhand++;
	}

	glPopMatrix();

	glPushMatrix();			//draw right sterno and right collar

	glTranslatef((*person)->r_sterno.xPos,(*person)->
r_sterno.yPos,(*person)->r_sterno.zPos);
	glRotatef((*person)->r_sterno.xRot,1,0,0);
	glRotatef((*person)->r_sterno.yRot,0,1,0);
	glRotatef((*person)->r_sterno.zRot,0,0,1);

	/*while(rsterno != (*person)->r_sterno.faces.end())
	{
		rsterno->draw();
		rsterno++;
	}
	while(rcollar != (*person)->r_collar.faces.end())
	{
		rcollar->draw();
		rcollar++;
	}*/
	glPopMatrix();

	glPushMatrix();			//draw right shoulder and right upper arm

	glTranslatef((*person)->r_shoulder.xPos,(*person)->
r_shoulder.yPos,(*person)->r_shoulder.zPos);
	glRotatef((*person)->r_shoulder.xRot,1,0,0);
	glRotatef((*person)->r_shoulder.yRot,0,1,0);
	glRotatef((*person)->r_shoulder.zRot,0,0,1);

	while(rshoulder != (*person)->r_shoulder.faces.end())
	{
		rshoulder->draw();
		rshoulder++;
	}
	while(ruarm != (*person)->r_upperarm.faces.end())
	{
		ruarm->draw();
		ruarm++;
	}

	glPopMatrix();

	glPushMatrix();			//draw right elbow and right hand

	glTranslatef((*person)->r_elbow.xPos,(*person)->
r_elbow.yPos,(*person)->r_elbow.zPos);
	glRotatef((*person)->r_elbow.zRot,0,0,1);
	glRotatef((*person)->r_elbow.xRot,1,0,0);
	glRotatef((*person)->r_elbow.yRot,0,1,0);

	while(relbow != (*person)->r_elbow.faces.end())
	{
		relbow->draw();
		relbow++;
	}
	while(rarm != (*person)->r_lowerarm.faces.end())
	{
		rarm->draw();
		rarm++;
	}

	glPopMatrix();

	glPushMatrix();			//draw right wrist and right hand

	glTranslatef((*person)->r_wrist.xPos,(*person)->
r_wrist.yPos,(*person)->r_wrist.zPos);
	glRotatef((*person)->r_wrist.xRot,1,0,0);
	glRotatef((*person)->r_wrist.yRot,0,1,0);
	glRotatef((*person)->r_wrist.zRot,0,0,1);

	while(rwrist != (*person)->r_wrist.faces.end())
	{
		rwrist->draw();
		rwrist++;
	}
	while(rhand != (*person)->r_hand.faces.end())
	{
		rhand->draw();
		rhand++;
	}

	glPopMatrix();

	glPushMatrix();			//draw root and torso

	glTranslatef((*person)->root.xPos,(*person)->root.yPos,(*person)->
root.zPos);
	glRotatef((*person)->root.zRot,0,0,1);
	glRotatef((*person)->root.xRot,1,0,0);
	glRotatef((*person)->root.yRot,0,1,0);

	/*while(proot != (*person)->root.faces.end())
	{
		proot->draw();
		proot++;
	}
	*/

	while(ptorso != (*person)->torso.faces.end())
	{
		ptorso->draw();
		ptorso++;
	}

	glPopMatrix();

	glPushMatrix();			//draw sacroilliac and pelvis

	glTranslatef((*person)->sacroilliac.xPos,(*person)->
sacroilliac.yPos,(*person)->sacroilliac.zPos);
	glRotatef((*person)->sacroilliac.zRot,0,0,1);
	glRotatef((*person)->sacroilliac.xRot,1,0,0);
	glRotatef((*person)->sacroilliac.yRot,0,1,0);

	/*while(sacro != (*person)->sacroilliac.faces.end())
	{
		sacro->draw();
		sacro++;
	}*/
	while(pp != (*person)->pelvis.faces.end())
	{
		pp->draw();
		pp++;
	}

	glPopMatrix();

	glPushMatrix();			//draw left hip and left thigh

	glTranslatef((*person)->l_hip.xPos,(*person)->l_hip.yPos,(*person)->
l_hip.zPos);
	glRotatef((*person)->l_hip.zRot,0,0,1);
	glRotatef((*person)->l_hip.xRot,1,0,0);
	glRotatef((*person)->l_hip.yRot,0,1,0);

	/*while(lhip != (*person)->l_hip.faces.end())
	{
		lhip->draw();
		lhip++;
	}*/
	while(lthigh != (*person)->l_thigh.faces.end())
	{
		lthigh->draw();
		lthigh++;
	}

	glPopMatrix();

	glPushMatrix();			//draw left knee and left calf


	glTranslatef((*person)->l_knee.xPos,(*person)->l_knee.yPos,(*person)->
l_knee.zPos);
	glRotatef((*person)->l_knee.zRot,0,0,1);
	glRotatef((*person)->l_knee.xRot,1,0,0);
	glRotatef((*person)->l_knee.yRot,0,1,0);

	while(lknee != (*person)->l_knee.faces.end())
	{
		lknee->draw();
		lknee++;
	}
	while(lcalf != (*person)->l_calf.faces.end())
	{
		lcalf->draw();
		lcalf++;
	}

	glPopMatrix();

	glPushMatrix();			//draw right hip and right thigh

	glTranslatef((*person)->r_hip.xPos,(*person)->r_hip.yPos,(*person)->
r_hip.zPos);
	glRotatef((*person)->r_hip.zRot,0,0,1);
	glRotatef((*person)->r_hip.xRot,1,0,0);
	glRotatef((*person)->r_hip.yRot,0,1,0);

	/*while(rhip != (*person)->r_hip.faces.end())
	{
		rhip->draw();
		rhip++;
	}*/
	while(rthigh != (*person)->r_thigh.faces.end())
	{
		rthigh->draw();
		rthigh++;
	}

	glPopMatrix();

	glPushMatrix();		//draw right knee and right calf

	glTranslatef((*person)->r_knee.xPos,(*person)->r_knee.yPos,(*person)->
r_knee.zPos);
	glRotatef((*person)->r_knee.zRot,0,0,1);
	glRotatef((*person)->r_knee.xRot,1,0,0);
	glRotatef((*person)->r_knee.yRot,0,1,0);

	while(rknee != (*person)->r_knee.faces.end())
	{
		rknee->draw();
		rknee++;
	}
	while(rcalf != (*person)->r_calf.faces.end())
	{
		rcalf->draw();
		rcalf++;
	}

	glPopMatrix();
	glPushMatrix();		//draw left ankle and left foot

	glTranslatef((*person)->l_ankle.xPos,(*person)->
l_ankle.yPos,(*person)->l_ankle.zPos);
	glRotatef((*person)->l_ankle.zRot,0,0,1);
	glRotatef((*person)->l_ankle.xRot,1,0,0);
	glRotatef((*person)->l_ankle.yRot,0,1,0);

	/*while(lankle != (*person)->l_ankle.faces.end())
	{
		lankle->draw();
		lankle++;
	}*/
	while(lfeet != (*person)->l_feet.faces.end())
	{
		lfeet->draw();
		lfeet++;
	}

	glPopMatrix();

	glPushMatrix();		//draw right ankle and right foot

	glTranslatef((*person)->r_ankle.xPos,(*person)->
r_ankle.yPos,(*person)->r_ankle.zPos);

	glRotatef((*person)->r_ankle.zRot,0,0,1);
	glRotatef((*person)->r_ankle.xRot,1,0,0);
	glRotatef((*person)->r_ankle.yRot,0,1,0);

	/*while(rankle != (*person)->r_ankle.faces.end())
	{
		rankle->draw();
		rankle++;
	}*/
	while(rfeet != (*person)->r_feet.faces.end())
	{
		rfeet->draw();
		rfeet++;
	}
	glPopMatrix();

	person++;			// push the pointer to the next avatar on
```

```
                                  // the list
}

   glPopMatrix();
}
float cubesApp::vec_length(Vertex v3)          //calculate the length between
{      float sum;                              //two points
       sum=v3.x*v3.x+v3.y*v3.y+v3.z*v3.z;      //sum=x^2*y^2*z^2
       return sum;
}
Vertex cubesApp::unit_vec(Vertex v1, Vertex v2)//convert a vector into a
                                               //unit vector
{      Vertex v3;
       float sum;
       v3=subtract(v1,v2);
       sum=vec_length(v3);
       v3.x=v3.x/sum;
       v3.y=v3.y/sum;
       v3.z=v3.z/sum;
       return v3;
}

Vertex cubesApp::subtract(Vertex v1, Vertex v2)//subtract vector v2 from
                                               //vector v1
       {
              Vertex v3;
              v3.x=v1.x-v2.x;
              v3.y=v1.y-v2.y;
              v3.z=v1.z-v2.z;
              return v3;
       }

void cubesApp::initGLState()
{
   GLfloat light0_ambient[] = { .2,  .2,  .2,  1.0};
   GLfloat light0_diffuse[] = { 1.0,  1.0,  1.0,  1.0};
   GLfloat light0_specular[] = { 1.0,  1.0,  1.0,  1.0};
   GLfloat light0_position[] = {2000.0, 1000.0, 100.0, 1.0};
   GLfloat light0_direction[] = {-100, -100.0, -100.0};

   GLfloat mat_ambient[] = { 0.7, 0.7,  0.7,  1.0};
   GLfloat mat_diffuse[] = { 1.0,  0.5,  0.8,  1.0};
   GLfloat mat_specular[] = { 1.0,  1.0,  1.0,  1.0};
   GLfloat mat_shininess[] = { 50.0};
   GLfloat no_mat[] = { 0.0,  0.0,  0.0,  1.0};

   glLightfv(GL_LIGHT0, GL_AMBIENT,  light0_ambient);
   glLightfv(GL_LIGHT0, GL_DIFFUSE,  light0_diffuse);
   glLightfv(GL_LIGHT0, GL_SPECULAR,  light0_specular);
   glLightfv(GL_LIGHT0, GL_POSITION,  light0_position);
   glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, light0_direction);

   glMaterialfv( GL_FRONT, GL_AMBIENT, mat_ambient );
   glMaterialfv( GL_FRONT,  GL_DIFFUSE, mat_diffuse );
   glMaterialfv( GL_FRONT, GL_SPECULAR, mat_specular );
   glMaterialfv( GL_FRONT,  GL_SHININESS, mat_shininess );
   glMaterialfv( GL_FRONT,  GL_EMISSION, no_mat);

   glEnable(GL_DEPTH_TEST);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_COLOR_MATERIAL);
   glShadeModel(GL_SMOOTH);
   glEnable(GL_BLEND);
   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
   glMatrixMode(GL_MODELVIEW);
}
```

```
int dotproduct(Vertex v1, Vertex v2){          //calculate the dot product of
       float result;                           //vector v1 and vector v2
       int ret;

       result = (v1.x * v2.x) + (v1.y * v2.y) + (v1.z * v2.z);
       if(result>0)
              ret=1;
       else
              ret = 0;
       return ret;
}

float hexdec(char a, char b)            //convert hexadecimal to decimal
{                                       //number
    float dec,c,d;

    if(a>'9')
       c = a - 'A' + 10;
    else
       c = a - '0';

    if(b>'9')
       d = b - 'A' + 10;
    else
       d = b - '0';

    dec = (c*16 + d)/256;

    return dec;
}



void cubesApp::read_file(char* file, list<Triangle>& tl)
{
       float c1[3], c2[3], c3[3];       // arrays which read in the
                                        //coordinates of vertices
       string rgb;                      // string which read in the colour
                                        //string
       Vertex ver_1, ver_2, ver_3;      // 3 vertices
       Color tri_color;                 // colour of the triangle
       Triangle tri;                    // create a triangle

    ifstream inFile;
    inFile.open(file);                         // Open file

       // Exit when the file cannot be opened
    if(!inFile)
    {
              cout<<"Cannot open file!!!"<<endl;
    }

       // Read each triangle from file and push it in the list
    do
    {
              inFile>>c1[0]>>c1[1]>>c1[2];      // read vertex one
              inFile>>c2[0]>>c2[1]>>c2[2];      // read vertex two
              inFile>>c3[0]>>c3[1]>>c3[2];      // read vertex three
              inFile>>rgb;                             // read colour

              // Change the colour string which to r,g,b colours
              float temp1 = hexdec((char)rgb[2], (char)rgb[3]);
              float temp2 = hexdec((char)rgb[4], (char)rgb[5]);
              float temp3 = hexdec((char)rgb[6], (char)rgb[7]);

              // Put coordinates in each vertices
              ver_1.input(c1[0],c1[1],c1[2]);
```

```
            ver_2.input(c2[0],c2[1],c2[2]);
            ver_3.input(c3[0],c3[1],c3[2]);

            // Set the colour of the triangle
            tri_color.input(temp1, temp2, temp3);

            // Input 3 vertices and a colour to the triangle
            tri.input(ver_1, ver_2, ver_3, tri_color);

            // Push the triangle to the back of the list
            tl.push_back(tri);

    }while(!inFile.eof());
}
```

10. Appendix C- Tabulated Results

group 1 = tube then bar			group 2 = bar then tube															
ID	group	gender	status	VRbefore	computeruse	beforeSAD	comfortable	responseToPeople	presence	talking	comms	awarepeople	impression	friendly	interested	meetagain	socialPerformance	social
1	1	1	2	4	7	14	3	2	4	2	3	3	4	3	3	3	20	2
1							4	4	5	5	4	5	5	3	3	5	60	0
2	2	2	2	5	7	4	4	3	4	2	1	4	4	3	2	3	50	0
2							5	3	5	3	1	3	5	5	4	4	70	0
3	2	2	3	1	7	1	5	5	5	2	3	5	4	4	3	2	60	2
3							5	5	6	3	3	5	4	2	2	2	50	0
4	1	1	2	3	7	11	5	5	3	3	2	3	4	3	1	1	30	0
4							5	6	5	5	3	6	6	3	2	5	60	0
5	1	2	4	2	2	17	2	3	4	1	1	4	2	5	2	3	1	4
5							2	1	3	1	1	4	2	5	1	3	3	3
6	2	1	3	5	7	12	3	5	5	5	6	6	3	3	2	3	50	3
6							7	7	5	1	4	4	4	5	5	2	80	0
7	1	2	2	4	5	7	5	4	6	4	4	5	4	6	5	4	80	0
7							3	4	6	5	5	6	3	3	3	2	60	2
8	2	2	2	1	6	7	5	7	1	2	2	6	3	2	2	2	80	3
8							7	7	1	1	2	3	5	3	3	2	85	1
9	1	1	2	3	7	1	3	4	5	1	1	5	4	5	1	1	100	1
9							5	5	6	5	5	5	3	2	3	2	80	1
10	2	1	2	1	7	1	4	5	3	3	2	5	4	2	4	3	60	2
10							2	4	2	1	2	3	4	1	1	1	60	0