Virtual Reality Platform Designed for Spatial Cognition Studies

by:

Ahmad Byagowi

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirement of the degree of
DOCTOR OF PHILOSOPHY

Faculty of Engineering
Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg

Abstract

In this dissertation, a new method of interaction with virtual reality (VR) has been presented. The VR environment, developed in this work provides a realistic and immersive environment that is developed for studying human's spatial cognition abilities. Different technical aspects of the implementation are described. To improve the accuracy of the experiments, a new method for comparing and analyzing trajectories has been presented. A set of human subject experiments have been conducted using the developed platform. The first experiment was based on an ordinary computer, a joystick and 40 cognitively healthy participants within the age range of 19 to 82 years. Participants were asked to perform the VR spatial cognition test by navigating in the VR environment. It was found that computer usage skills had an effect on the overall performance of the participants as well as motion sickness; such an effect was undesirable. To resolve both problems, a novel input device made from the combination of a wheelchair and a motion capture unit, called VRNChair was developed. The VRNChair allows the participant to move physically in the real world in order to navigate in the VR environment. This allows participants to have a more intuitive interaction with the VR environment. Since the participant physically experiences the motion seen in the VR, motion sickness is minimized. A test based on two age groups, 34 young (< 40 years) and 20 older (60+ years) participants was performed. The results showed no difference between using the VRNChair or the joystick for the young participants while it showed considerable improvement among the older participants. Furthermore, it was noted that participants became distracted while using the VRNChair, Hence, a head mounted display (HMD) was added to the platform. A test with 14 males showed less distraction to the participant by using the HMD. Overall, the VR environment designed in this work passed all the validation experiments successfully. It offers a novel solution to overcome induced motion sickness experienced as a side effect of VR environments.

Aknowledgments

Hereby, I want to thank my advisor Prof. Zahra Moussavi for her support and help during my education. She provided me the best opportunity to explore more about human's brain and its capabilities. I was able to work in a field that research was no longer anything other than my own curiosity to discover my surrounding world. In addition, I would like acknowledge Prof. Witold Kinsner for his support and guidance which kept me inspired and willing to continue this new stage of my life. I would also, like to acknowledge Prof. Robert McLeod for letting me discover a new world in computer scince. Similarly, I would like to acknowledge Prof. Pourang Irani in showing me a the bridge between humans and computers. Moreover, I would like to thank the technicians at the Department of Electrical and Computer Engineering, namely Glen Kolansky, Zoran Trajkosky and Sinisa Janjic for all their support throughout all my educational years.

I like to thank all the recruitment team and the volunteers for making the experiements possible. Moreover, I like to thank all the summer students and my lab partners for their help and support. Additioanlly, I would like to thank my UMSATS team namely, Mike Lambeta, Paul White, Pawel Glowacki, Greg Linton, Kazushige Kimora, and Jose Juan Mijares Chan for their help and support.

Finally, my most profound thanks are to my father, who is the inspiration for my life and my mother, who has always given me her infinite love and devotion, my sister Shima, my brother Ebrahim, Wendy, Joel and Jesslyn Denae Janssen for their love, understanding, support and trust.

Dedicated to:

Peter Kopáček and my parents

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AGM	Accelerometer Gyroscope Magnetometer
ANC	Acclaimed Noise Cancellation
AR	Augmented Reality
BVH	Bounding Volume Hierarchy
DK	Development Kit
DOF	Degree of Freedom
FPS	First Person Shooter
GVS	Galvanic Stimulation
HD	High Definition
HID	Human Interface Device
HMD	Head Mounted Display
IMU	Inertial Measurement Unit
IPD	Interpapillary Distance
IR	Infra-Red
LCD	Liquid Crystal Display
MDI	Minimum Distance of Interest
MEMS	Micro Electro Mechanical Systems
MoCA	Montreal Cognitive Assessment
QC	Quiet Comfort
RAM	Random Access Memory
RDW	Redirected Walking
TBI	Traumatic Brain Injury
VR	Virtual Reality
VRNChair	Virtual Reality Navigation Chair

Chapter 1

Introduction

"The knowledge of anything, since all things have causes, is not acquired or complete unless it is known by its causes"

- **Avicenna**

In the past decade, due to the advancement of computer power and technology, virtual reality (VR) experiments have gained popularity. A VR environment allows for natural world emulation for different purposes by replicating the physical presence of the real world. In such simulations, certain tasks can be performed virtually. Experiments in a VR environment offer repeatability and flexibility to modify the testing environment. VR can be used in different fields such as teaching [1], medical surgical training [2], neuropsychology [3], sports [4], interior design [5], military and countless other areas [6]. Moreover, monitoring movement parameters such as position and trajectories of a participant in a natural environment is sophisticated, whereas in a VR environment it is relatively simple [7].

Due to advances in computer technologies such as processing power, hardware miniaturization, advanced computer displays and embedded graphic processors, VR became more accessible and possible to the end users at a lower cost [8]. Technologies such as head mounted displays (HMD), high definition liquid crystal displays and inertial measurement units made it possible for

manufacturers (such as Oculus) to produce low-cost consumer-grade head mounted displays suitable for VR purposes [9].

In art, realism is defined as the general attempt to depict subject matter truthfully [10]. In VR simulation, realism is the attempt to make the illusion of a real physical presence in the synthesized environment. There are two main concepts that drive VR towards realism: immersion and interactivity; these two complement each other. Immersion aims to minimize the distractions of the real world to achieve a selective focus on the information presented to the user [6]. An immersive VR environment makes use of sensory stimulation and affective computing (senses the user's reactions and emotion) [11] in order to increase a sense of presence. Interactivity denotes the user's level of connection and involvement with the virtual environment; this aspect can be increased using both intuitive interaction and naturalistic (imitate or reproduce nature in effect) computer graphics (auditory, tactile and other simulations may also be added).

The design of a VR environment requires a computer system comprised of hardware and software, and the knowledge of human perception about distance, time and objects [12]. The recent improvements of current VR systems in hardware and software have allowed for a more intuitive interaction for humans in VR. For instance, 3D navigation through a VR environment was initially achieved using arrow keys in a desktop computer [6] [13]. The use of the joystick [14] enhanced the interactivity of the user with the computer and allowed for more realism. In addition, there have been improvements in terms of providing higher image resolutions, rendering and delivery of higher frame rates. Overall, a VR engine should be easy to modify in order to create different settings and environments as required.

1.1. Motivation

The main motivation of this thesis was to design and implement a VR environment that can help in spatial cognition assessment, particularly for a wide age range and more importantly for older people with little to no computer skills. Spatial cognition requires knowledge and mental representation of real world objects in order to convert the geometric properties of objects and their spatial relationships [15]. Thus, our desired experiments concentrate on the self-to-object spatial cognition of the subject. Spatial cognition can be best assessed by the accuracy of the subject's navigation to reach a specified target throughout an environment.

The use of a physical environment for studying spatial cognition is challenging for two main reasons: one, providing a large environment without landmarks is not feasible for a natural setting, and two, control and reconfiguring objects in a natural environment is very difficult. Therefore, repeatability and the reliability of the experiment would be questionable. For that reason, many researchers make use of maps [16] [17] and other visual representations such as videos to evaluate the navigation skills and spatial cognition of subjects [18]. The use of these methods may provide repeatability but they do not confer a sense of reality; in addition, simulation based experiments may not be suitable for older participants. Thus, interpretation of the results and their accuracy in representing spatial cognition remains a challenge.

The use of VR as an alternative for traditional non-computer based spatial cognition experiments comes with several benefits such as flexibility, repeatability and accuracy of the experiment as mentioned above. VR has the potential to resemble any anticipated setup without the need of using a real place or map, keeping a certain sense of reality. Moreover, VR can help to acquire and process the data from each subject more easily and more accurately. However,

conventional immersive navigational VR environments are limited to a chamber or theater (a CAVE-like installation [13]) equipped with motion tracking sensors [19]. Such systems may be costly and may require a specialized space dedicated exclusively for that purpose which may not be feasible and affordable everywhere.

To achieve a high level of realism, a VR environment should provide a high level of immersion to the participant by simulating the real world physics as much as possible. Nevertheless, there is a compromise between the detailed depth of the simulation and the computer processing demand. The hardware limitations should also be considered when designing a naturalistic simulation of a VR environment; otherwise, the VR environment will have differences with the real world experience. In addition, interaction between the user and the VR environment differs from that between the user and the real world. Therefore, the VR simulator should be able to interact with the user in an intuitive manner. For example, using ordinary computer input and output devices may introduce a bias relative to the level of the experience of the user, particularly for older adults with limited computer experience. Therefore, an intuitive and natural input device has to be sought to reduce this bias.

The computer and game entertainment industry has scaled up the demand for more advanced graphics and greater hardware processing abilities. Computer simulations try to imitate the laws of physics in a naturalistic and reliable manner. In order to reduce the high costs of producing a single naturalistic game, companies have come up with modular game engines that can be used for developing similar games [20]. Thus, the possibility to implement the latest advancements in gaming technology was opened up to the public and researchers without the need for expensive workstations [20]. To date, games such as first person shooters make use of the latest technological

advancements to deliver some of the most naturalistic and most immersive games available. The use of a game engine can also be valuable in the design of a VR environment for research purposes.

For the purpose of human's spatial cognition, after successfully creating a naturalistic VR environment, the user must be able to navigate within it. However, walking in a very naturalistic VR world has presented several issues. One of them is motion sickness or more specifically, kinetosis [21]. Some users, particularly those of age 35+ years, experience it due to the lack of physical displacement of their bodies. As a result, several VR systems have tried to resolve this problem by using modified treadmills (e.g. omnidirectional treadmill) [22] that provide an infinite surface [23]. However, that solution does not provide physical input during rotation and still can cause motion sickness if the navigation requires several rotations; in addition, the user may experience a superficial level of immersion due to the instability and the differences in walking on a treadmill compared to walking in the real world. Hence, motion sickness hinders the level of immersion of the user, and in some cases, it incapacitates the user's ability to navigate through the VR environment; this problem is thoroughly addressed in this study.

Once a VR simulator satisfies the previously mentioned conditions, an evaluation that compares the real world against the VR environment should be performed. The validation of the usefulness of the VR-based experiment for studying human spatial cognition must compare people's performance in both virtual and real worlds with identical environments. Such an experiment can be performed by rendering the real world into the virtual world in real-time. This will allow the participant to navigate through the world by using VR feedback exclusively. The use of this method can be considered as an alternative and subsequent replacement of the real world once it is compared to an experiment conducted in the real world; this is the objective of the future work of this study.

1.2. Problem Statement

Studying spatial cognition requires a test environment. Construction of a reconfigurable physical test environment can be very challenging. A naturalistic VR platform offers a solution for this problem. Thus, the challenge would then be to increase the immersion and naturalistic aspects of the VR environment without causing any motion sickness. Additionally, interaction with a VR platform needs to be as intuitive as possible to allow the experiment to map the participant's performance in VR, similar to how they perform in the real world.

The research questions are:

- Can a VR platform be used to study navigational spatial cognition?
- How can the fidelity of a VR based spatial navigation experiment be maximized?
- How can the motion sickness and computer user experience bias be minimized?

1.3. Objectives

The short-term objectives of this thesis have been to:

- Investigate the potential of virtual reality to study spatial cognition
- Improve human computer interaction using a novel input device
- Compare user experience between different VR methods
- Increase the immersion by using an HMD and address its challenges

The long-term objectives of this thesis have been to:

- Develop a tool to support navigational spatial cognition studies

- Reduce unwanted computer-usage-experience-bias
- Reduce motion sickness induced from VR-based experiments
- Provide a mathematical framework for the VR environment designs and performance

1.4. Thesis Organization

This thesis is divided into six chapters. Following this introduction, chapter 2 gives a background and discusses the cutting edge of VR. It starts with a history about video games and their evolution over years. This is then followed with the development of game engines and a description about the technical aspects of VR. It continues with the description of VR continuum. The VR continuum is a scale to classify a system from completely real to completely virtual [24]. Chapter 3 describes the theory behind the VR engine, starting with the overall structure of the game engine and the visual rendering, followed by audio rendering and the simulation of a selection of physics laws in the VR environment. Then the theoretical background used to design the VRNChair is presented, and followed by the artificial intelligence and the data analysis method used in this work to analyze the results obtained from the experiments. Chapter 4 describes the design and implementation of the experiments used in this study. The results and discussion are presented in Chapter 5. Chapter 6 presents the conclusion and recommendations for future work.

1.5. Summary

With the advancement of computers in the past decades, VR platforms became more realistic and popular for both gaming and neuro-cognitive studies. A large body of spatial cognition studies focus on spatial orientation during navigation to understand how the brain encode environmental features to navigate towards a destination. These studies are experimental based and due to lack of navigational VR to date, they have been limited to studies in small sizes so that it could be run in real environments. Development of a VR environment to allow physical navigation as a replacement for real experimental environment would be very advantageous as it will allow to run a study with reconfigurable environment while controlling all desired variables, where its real replica is not plausible due to cost and space of having similar reconfigurable physical environment.

Chapter 2

Background

This chapter aims to give a background about video games, their history, how they became popular and different classifications. It is followed by the concept of game engines, where parts of the game software are reused to reduce the development time of other games. Subsequently, the concept of virtual reality (VR) is explained by its history and recent enhancements, and how VR benefited from the recent computer hardware advancements. Finally, the concept of VR continuum is described that attempts to put reality and virtuality on a continuous spectrum.

2.1. Video Games

The cathode ray tube amusement device, introduced in 1947, is known by most as the first video game [25]. It took a while until video games became more popular around the 1970s and 1980s when arcade games, gaming consoles and home computer games were introduced to the public [26]. Since then, video games became more popular to the point that they are part of today's life for a considerable portion of the society. Today, the video game industry is an undeniable entertainment market with a generated annual sale of approximately $75B [27].

As the video games became more popular, the demand for more complex and capable games increased. By the year 1984 the first attempt to classify video games in different genres lead to the introduction of several classes of games [28]. Video games were no longer just two paddles and a bouncing ball (such in the PONG game), trying to resemble a 2D ping pong game. Games started

to become more naturalistic and more capable of engaging the player within the game [29]. As video games started to become more advanced, they started to be more hardware demanding. Different manufacturers started to present hardware platforms, specifically designed for modern video games [30]. Some of the game producers tried to offer multi-platform titles. Developing games for different platform requires specific designed pieces of software for individual platforms. The rest was reusable for any platform [31].

Beside different genres, games are classified based on their purpose and objectives. Some of the classes are casual, educational or serious games [32]. Serious games are those that are not solely for entertainment purposes such as giving the user a learning experience of some sort. This class of games can also be used to test the user's ability to perform a certain task [33].

2.2. Game Engines

Before the introduction of the game engine concept, games were developed as singular entities. While an advanced video game is developed for a specific title, parts of it can be reusable for other games. This concept can be advantageous to reduce the development time for a game [31]. A collection of the reusable parts of a game and development and support tools is called a game engine [34]. Usually a game engine is designed for, if not general, a wide range of game classes and types. However, some game engines are known to be more efficient and can provide better features for certain classes of games. This is due to the trade-offs that are considered during the development of each specific game engine [35].

The term "game engine" was introduced around the mid 1990s. This is the time where first person shooter games were introduced; games such as Doom, and Quake which were essentially a

simulation of a first person view navigating in an environment shooting monsters [36]. Doom and Quake games were written as two separate pieces: the game engine and the content repository [35]. This division allowed other developers to focus on the contents and the game rather than technicalities and low-level software challenges.

Using an off-the-shelf game engine can speed up the development of a game. However, without access to the source code and deeper understanding about the low-level implementations of the game engine, the developer is restricted to the features offered by the game engine. In addition, a more general purpose game engine is not optimized for a specific task; therefore, specific applications with a high demand in certain aspects, such as naturalistic VR may suffer from the offered features.

Naturalistic games require game engines that can simulate the physical presence with high fidelity in term of visual and sound synthesis. Additionally, they should be able to mimic certain laws of physics, mainly rigid body dynamics, to appear and feel more real and create a higher immersion for the user [37]. This task can be performed seamlessly inside a game engine or using a middleware physics engine [38].

2.3. History of Virtual Reality

The dawn of VR commenced in 1965 when Ivan Sutherland envisioned the “Ultimate Display”; The display showed a computer-rendered space or image as if it was real; he called it “Virtual World”. He obtained this term from Suzanne K. Langer’s writings on philosophy of aesthetics [20]. However, the term “Virtual Reality” was coined by computer scientist Jaron Lanier co-founder of VPL Research, the first firm to sell VR equipment [21]. Since the beginning, he always

envisioned VR for tele-immersion, which is a fusion of video conferencing and VR [20]. Despite his efforts on research over high speed-networks, VR has not been embraced in the society as other technologies have gained popularity.

Development and design of VR is an art to mimic some of the nature's phenomenon. It makes use of a 3D perspective, fractal, and pseudo-color image processing [22]. Since then, VR has had its greatest impacts on military and flight simulator applications. As well, it has grown slowly into the entertainment and game industry that has made the greatest leaps to push this technology forward. Thus, this technology has advanced into two different branches: the work on VR applied to a particular profession or commercial product and the development of VR to perfect the technology itself [1]. However, VR has undergone a much greater development on the first branch due to the economic and social impact.

One of the first applications of computer-based VR was on vehicle simulations such as flight simulators for training and military purposes. The early developments were costly and involved cockpit avionics and pilot controls mounted on large structures moved by hydraulic platforms. Nowadays, cheaper systems are being created with head mounted display (HMD), joysticks and data gloves [23]. Most recently, the military area has benefited from this technology by using it to evaluate brain disorders in active duty soldiers such as traumatic brain injury (TBI) [24]. Psychological tests performed using VR in this population better replicate natural conditions and thus improve detection on neurocognitive deficits in military settings [24].

One of the classic examples of military and entertainment (gaming) applications of VR are first person shooters (FPS) [9] [10]. These fields have provided the latest advancements due to high consumer demand. FPS simulators made a leap from only using a keyboard, to a mouse and a

keyboard, to a joystick or gamepad. FPS games have the characteristic of putting the player in a first person perspective (as if it was seen through their own eyes) and display an avatar's hands carrying a weapon. As a result, the gaming industry created more intuitive games and motion controllers that have triggered an advance in the VR field as in the Nintendo® Wii remote [25], the Sony PlayStation 3 Move motion controller [26] and the Microsoft® Kinect sensor [27], which focus on human-computer interfaces that target a wider audience [26]. Among them, the Wii remote has delivered the greatest interactivity because it can be held as a real weapon and the player can point and shoot (towards the sensor mounted on top of the monitor or TV) as if it were a real gun.

However, the newest trends on VR are on the area of medical surgical training and rehabilitation. The introduction of minimally invasive surgery has placed huge challenges on the classical surgical training. As laparoscopic and endoscopic procedures became more common, they became associated with a higher rate of complications [28]. The operating room is not a suitable learning setting because of medico-legal issues, pressure and costs. Consequently, VR has become an alternative to train students as it can easily replicate the intervention using similar hardware and a simulation software that mimics the different procedures.

If the VR environment incorporates teaching resources and evaluation metrics, it can help students to learn the level of expectation required [29]. It has been shown that surgeons that are computer game users make fewer errors [30]; this implies that VR can be of great use as a teaching tool. In addition, VR training with haptic feedback can help to improve the surgeon's psychomotor skills while getting an interactive feedback [31].

As VR has progressed through the medical field, it has reached medical rehabilitation, psychiatry, neuropsychology and functional and cognitive psychology. In areas such as medical rehabilitation, traditional rehabilitation is costly and constant modification for each patient is necessary [32]. Thus, VR can be helpful because the environment can be easily customized, and can be used at the patients' home when the therapist is not available and/or to reduce costs. Similarly, in the areas of mental health, this technology has had a great impact, and it is rapidly growing. Most of the VR applications in this field have tried to target the ageing process, assessment of sensory, mood and loyalty [39], motor abilities and phobia treatment [5]. The use of VR has indeed marked a new era in the study and treatment of psychological disorders.

Within the last decade, spatial cognition and the ability of navigation has been studied using the latest technologies to improve the accuracy and reality of the results. Traditional neuropsychological tests measure the ability to solve one single problem at a time, they are short and the examiner gives cues of whether the subject did a successful performance or not [5]. Besides, these tests use highly unusual stimuli and paradigms, which are non-ecological and non-representative. Immersive VR has been used to evaluate learning and memory by studying navigation using a city setting similar to an FPS where different threats were posed to the subject [33]. In addition, a neuropsychological test (the Wisconsin Card Scoring Test) has been replaced with a VR simulation of a building where clues similar to the card test, help one to navigate from room to room [5]. These improvements have revolutionized the field of psychology and have opened new doors to investigate human nature, behavior and their disorders.

The previous studies have led a new group of researchers to create potable VR systems for the outdoors [34]. One of these studies has used a GPS to walk through a field while carrying a laptop on a backpack and an HMD using the VR environment, this is called redirected walking (RDW)

[35]. RDW is carried out to measure the spatial navigation skills while walking outside; this allowed for a higher level of immersion [34]. However, this method is not accurate enough due to the limitations that GPS provides. Development of new technologies such as HMD's (e.g. Oculus Rift) [21] has created more immersive VR environments. The outcome, however, has caused simulator motion sickness for a considerable number of users, which is thought to be caused by the inconsistent information about the body orientation and motion received by other senses [36].

2.4. Virtual Reality Continuum

The advancement of the VR technology quickly led to a branching and diversification of the field and the technology itself. Hence, Milgram and Kishino, created a classification of the different subsets of VR into what they called VR continuum or Virtuality continuum as seen in Fig. 2.1 [40]. This continuous scale ranges between reality which is situated at the far left (the actual world), virtuality at the far right (a computer graphics generated world), and mixed reality lies in between, where both are combined [40] [41]. The different levels of mixed reality depend on the amount of reality or virtuality.

The resulting technologies try to incorporate features from reality such as objects as part of their interface with computer-generated images. The levels of the continuum are: reality, the world we live in; the next level is augmented reality (AR), which enhances the real world with computer generated objects that appear to occur in the same space as in the real world [42]. Although most people are unaware of it, AR has widely impacted everyday life as it is in the case of sports or races where images, annotations and stats are shown in real time [42]. In addition, AR has recently been used in the rear view cameras for parking aid systems [43]. Other applications for this

technology have been made in the medical field, and for collaborative, commercial or educational projects. Such has been the case with the MagicBook and the Studerstube that both encompass education and collaborative interaction [41] [44].

The opposite case is AR, which integrates real objects into a virtual environment but has not been as widely used as AR. One of the main uses of AR has been videoconferencing in a virtual world using the live video streams of the participants [45]. More work should be done to push forward this technology into more fields.

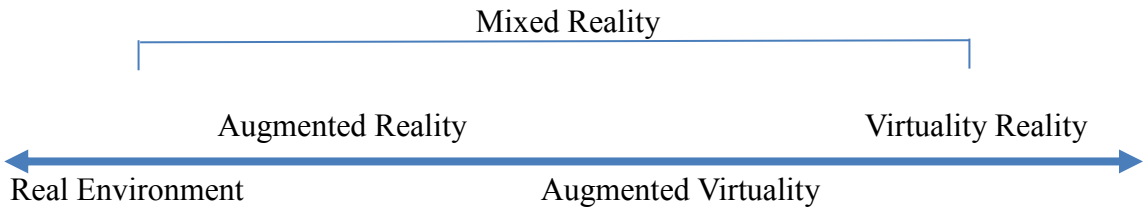


Fig. 2-1. Virtual reality continuum.

The VR continuum is not a discrete scale, as its name implies, many more types can occur in between them. Several sub-classifications have appeared within the mixed reality categories depending on the technology and the hardware used. There are cases that the amount of reality and virtuality can be adjusted to affect the perception of the user to experience the full spectrum of the continuum, as it is the case of the MagicBook [41].

Sometimes the level of immersion of VR may affect the perception of reality observed in the mixed reality environment. Therefore, enhancing the realism of VR can decrease the difference between real and virtual perception of the world. Moreover, the level of realism in simulation and the incorporation of the real object depend on the technology and the simulator software. These

can generate different levels of immersion for the observer depending on the quality of the devices used.

2.5. Head Mounted Displays

While the concept of HMD dates back to 1945 with a wearable TV patent [46], it was around the 1990s when wearable computers started to come to the consumer market [47]. Perhaps, it was due to the lack of technologies such as liquid crystal displays (LCD), inertial measurement units (IMU) and miniaturized microprocessors, which are directly involved in building a modern HMD [48].

One of the most popular consumer level HMD is the Oculus Rift [49]. It utilized a consumer level LCD and a low cost micro electro mechanical system (MEMS) based inertial measurement unit (IMU) for its first generation released (DK1) in 2012 [50] to keep it total price under $250. In 2014, the second generation (DK2) was released as an advancement to its predecessor with a higher resolution LCD (1080x1200) and infrared (IR) based tracking system. Today, the Oculus Rift is widely used by researchers, developers and gamers as an affordable HMD for VR and gaming purposes [51].

2.6. Summary

The advancement of video games led to the new structure of game specific contents and the reusable part, known as the game engine. A class of video games was developed based on the first person perspective and was capable to induce the illusion of presence. The class later developed into the VR. Aspects of VR can be described on a continuum between reality of virtuality based on the implementation.

Chapter 3

Method

In this chapter, the mathematics of developing VR platforms is derived and described. The virtual reality (VR) environment is a computer simulation of a physical presence in an environment. The simulation software of a VR design is called a game engine, which is a software framework entailing different parts [52]. This framework is comprised of various parts such an artificial intelligence component, visual rendering, sound rendering, physics engine, graphic and auditory assets, data collection, memory management, depending on the application [53].

3.1. Implementation of a Game Engine

The game engine can be implemented in different ways, and may include or exclude the parts mentioned above, based on the application. In this thesis, the implementation is based on two preliminary functions, setup and update functions as shown in Fig. 3.1 [54]. The setup function is executed once to perform initializations and the update function is executed periodically until the simulation is stopped by the exit function.

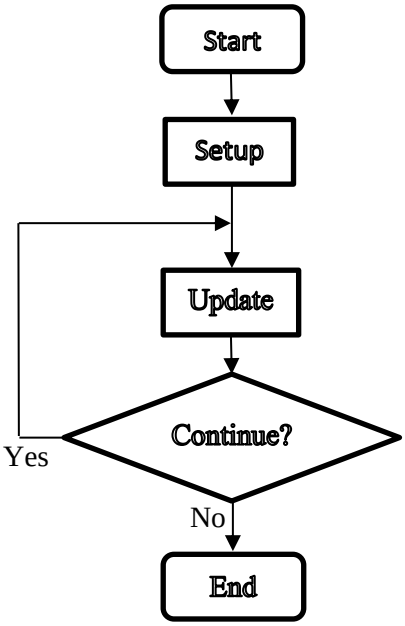


Fig. 3-1. Implementation of the game engine used in this work.

Setup function

The setup function is called only once at beginning when the simulation is being initiated. The setup function will include sub-functions that are needed to be executed at the beginning to perform initialization of the system. The sub functions are listed as follows:

1- Hardware resources verification

2- Memory allocation and initialization

3- Graphical asset loading

4- Auditory asset loading

5- Setup of the VR environment

6- Initialization of simulation parameters

7- Initialization of the artificial intelligence

8- Initialization of logging files

Update function

The update function runs periodically until the exit command is issued. The sub-functions are listed as:

1- Input acquisition

2- Physics engine

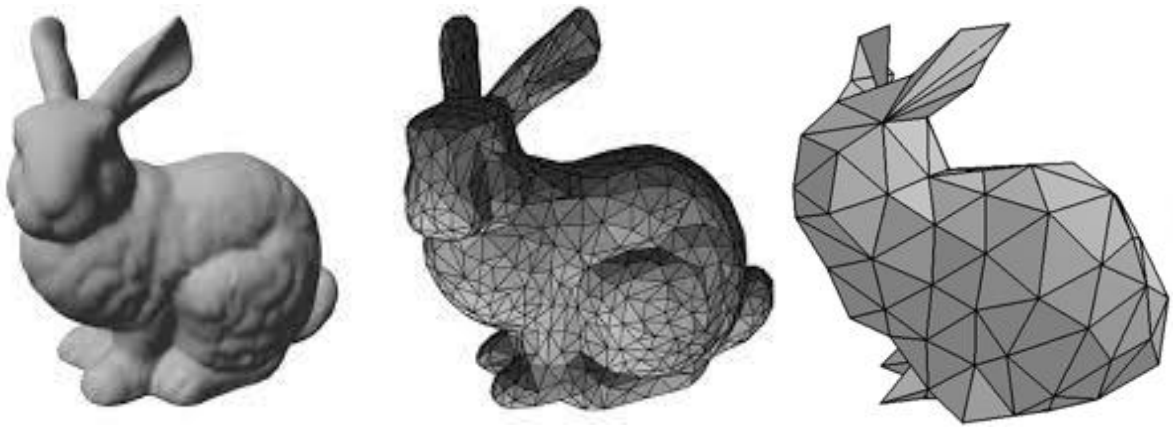
3- Visual rendering

4- Auditory rendering

5- Logging the simulation parameters

3.2. Visual Rendering

The visual rendering unit takes care of what is displayed on the monitor (or head mounted display) to visually resemble the VR environment. Essentially, a visual rendering engine is a real time graphic projection engine that is optimized to draw triangles in 3D and rasterizes the filling pixels with the consideration of the material [53]. In addition, to obtain naturalistic results from rendering it is important to simulate lighting effects on the rasterized filling pixels. Since the rendering engine is designed and optimized to draw triangles in 3D; every object should be modeled based on the requirements, solely using triangles. Figure 3.2 shows different levels of details in modeling the famous Stanford bunny 3D model. Different applications may require a different level of details. A trade-off between detail and required processing load makes choosing a level of details an important factor to keep the rendering processing load feasible for real time display.



(a) (b) (c)

Fig. 3-2. Stanford bunny shown different levels of details, (a) high, (b) medium and (c) low.

A collection of 3D models from different objects and an environment can constitute a VR environment.

3.2.1. Visual Construction of a VR Environment

The VR environment is constructed from 3D object and environmental models, which together are called the world model. A world model is composed off a set of object models, placed in defined locations and orientations. The world model is loaded upon initialization of the simulator into RAM. Moving in the VR environment is simulated using perspective projection. This comprises off translation and rotation about the three dimensions. The mathematical principles behind the perspective projection are explained in the following sections.

3.2.2. Perspective Projection

Let us consider the view plane $\overrightarrow{N}.\ \vec{X} = d$ and the eye point $\vec{E}$ in the positive side of the plane. This results in $\overrightarrow{N}.\ \vec{E} > d$. With the condition of $\overrightarrow{N}.\ \vec{E} > \overrightarrow{N}.\ \vec{X}$ the respective projection of point

$\vec{X}$ onto the projection plane is the intersection of the ray starting from point $\vec{E}$ that contains $\vec{X}$. If we consider the eye point the origin $\vec{E} = (0, 0, 0)$ and the view plane $z = n > 0$, the plane normal will be $\vec{N} = (0, 0, -1)$ with the plane constant $d = -n$ [53]. The projection can be described as $\vec{Y} = (1 - t)\vec{E} + t\vec{X}$ with the condition of $\overrightarrow{N}.\ \vec{Y} = d$. The value *t* can be found using

$$t = \frac{\overrightarrow{N}.\ \vec{E} - d}{\overrightarrow{N}.\ \vec{E} - \overrightarrow{N}.\ \vec{X}} \tag{3.1}$$

where it yields $t = n/z$. Since the projection plane remains fixed at $z = n$ the projected point of an arbitrary point in space described with (x, y, z) will be $(\frac{nx}{z}, \frac{ny}{z}, n)$ or simply as a 2-tuples $(\frac{nx}{z}, \frac{ny}{z})$. Additionally, the variable $w = \frac{z}{n}$ can be described so that the view plane is located on $w = 1$ and the projected point will be $(\frac{x}{w}, \frac{y}{w})$.

The 2D projection reflects a perspective view from 3D objects. Eliminating portions of the data that are not affecting (not visible in) the 2D projection can improve the overall rendering performance. This process is performed using culling and clipping. Culling is the process of determining if an object can be eliminated and clipping is the process of eliminating parts of an object for rendering.

A camera model is responsible for managing the asset contents and selects "what and how" to draw on the output medium. The asset contents are digitally designed 3D meshes and textures. In a camera model, managing "what and how" to draw the selected objects is based on 3D perspective representation of a perceiver in a physical world. The camera model clips a subspace (shaped like

a portion of a pyramid) from the entire game world, which is called the frustum volume. Then, the 3D volume is rendered, and projected into 2D data in screen space. This is called the perspective projection.

The VR environment is constructed from a collection of 3D meshes wrapped in corresponding textures. The VR environment is larger than what a camera is required to show the user at an instance. On the other hand, the processing required for rendering is relatively heavy. Therefore, reducing the rendering to objects that are within the field of view and not processing those objects that are outside the view volume (culling) can improve the rending processing load significantly. In addition, objects that are partially in the view volume (on the boundaries) can be partially rendered; this process is called clipping.

Rendering the view volume includes the projection onto the view plane $z = n > 0$. As shown in Fig. 3.3 the rectangular region of interest from the view plane is called the viewport.

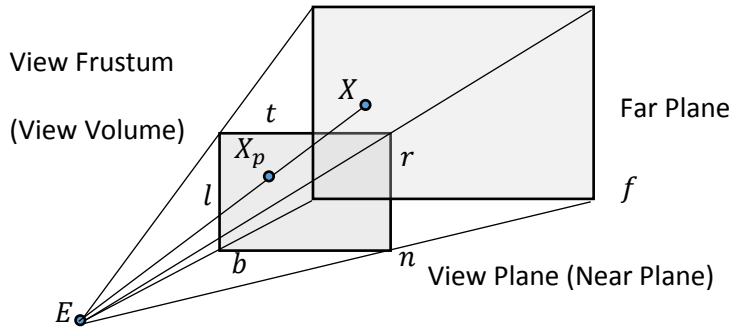


Fig. 3-3. Perspective projection and view frustum.

The view and near planes can be considered separately. This will make the effect of seeing a picture projected on a screen at a distance from the perceiver. Therefore, we assume the view plane is the same as the near plane, which will imply a higher degree of immersion for the perceiver.

The view volume is limited by the far plane $z = f > n$. The view plane is limited by four planes, top plane $y = \frac{tz}{n}$, bottom plane $y = \frac{bz}{n}$, left plane $x = \frac{lz}{n}$ and the right plane $x = \frac{rz}{n}$.

Based on the positioning of the camera on the point $facing$ toward the view plan, up direction will be defined using the vector $\vec{U} = (0, 1, 0)$ and left direction using vector $\vec{L} = (1, 0, 0)$. The viewport is defined as a rectangle with range of -1 to +1 for both its horizontal (x-axis) and vertical (y-axis), i.e., normalized x and y coordinates. The view plane needs to be scaled to accommodate the required range. The scaling factors are calculated using

$$S_x = \frac{2}{r-l}\left(x - \frac{(r+l)z}{2n}\right) \tag{3.2}$$

$$S_y = \frac{2}{t-b}\left(y - \frac{(t+b)z}{2n}\right) \tag{3.3}$$

The following projection can be implied using a homogeneous transformation derived from

$$H_{project} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{3.4}$$

The $H_{project}$ transformation assumes the eye to be at the origin facing the z-axis. In order to simulate moving in VR environment it is necessary to have six degrees of freedom (DOF), which are composed from three axes translation, and three axes rotations. All together, they are called the 6-DOF transformation. The camera can be moved and/or rotated to any arbitrary place using the following affine homogeneous transformation described as

$$H_{trans} = \begin{bmatrix} \begin{bmatrix} & Rotation & \end{bmatrix} & \begin{bmatrix} Translation \end{bmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} \tag{3.5}$$

The product of $H_{project}$ and H_{trans} can map any given position and orientation from the VR environment onto the normalized viewport.

3.2.3. Model Coordinates

Every object in the VR environment is described based on 3D meshes. 3D meshes are constructed from a set of triangles, quadrilaterals or convex polygons in 3D. In this work, the triangles are used to describe 3D meshes due to the relatively simple objects and lower processing requirements. Every triangle is described using three vertices. Each vertex is described using three x, y and z values based on the object's coordinate system. Figure 3.4 shows how a cube is described using triangles and further using vertices.

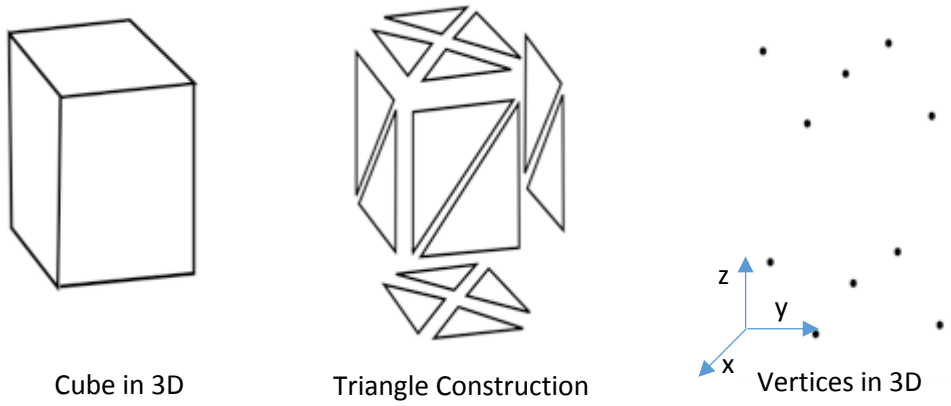


Fig. 3-4. The construction of a 3D object like a cube using triangles and vertices.

The VR environment is constructed from multiple 3D objects (3D meshes) in different locations, orientations and scales. One object can be used a single time or multiple times. The placement of the objects in the VR environment can be done using a specialized editor or a set of instructions. Whether the VR environment is constructed using an editor or the set of instructions, the outcome will be a collection of affine homogeneous transformations to place, the 3D objects in different locations, orientations and scales. The location and orientation transformation can be performed using the H_{object} matrix while the scaling can be applied in different axis with different or similar values (uniform scaling) using the S_v matrix [53]. The H_{object} and S_v matrices are written as

$$H_{object} = \begin{bmatrix} \begin{bmatrix} & Orientation & \end{bmatrix} & \begin{bmatrix} Location \end{bmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} \tag{3.6}$$

$$S_v = \begin{bmatrix} Scale_x & 0 & 0 & 0 \\ 0 & Scale_y & 0 & 0 \\ 0 & 0 & Scale_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(3.7)

In most cases, the scaling is performed equally on all the three axes, which is called uniform scaling. In the case of uniform scaling, the matrix S_v can be simplified and written as

$$S_{uniform} = \begin{bmatrix} Scale & 0 & 0 & 0 \\ 0 & Scale & 0 & 0 \\ 0 & 0 & Scale & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \dfrac{1}{Scale} \end{bmatrix}$$

(3.8)

This form of scaling can be applied directly to the location and rotation transformation using the combined matrix written as

$$HS_{object} = \begin{bmatrix} \begin{bmatrix} & Orientation & \end{bmatrix} & \begin{bmatrix} Location \end{bmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & \dfrac{1}{Scale} \end{bmatrix}$$

(3.9)

An object is made from a set of vertices described in the homogeneous form $[x, y, z, 1]^T$. Since, every object is described using a set of vertices, each vertex (P) should be placed in a specific location in VR environment. Based on the location of the observer in the VR environment, a set of vertices will be projected onto the viewport. The projected point (P') is calculated by

$$P' = H_{project} H_{trans} HS_{object} P$$

(3.10)

The viewport is two-dimensional while the projected point (P') is described in three dimensions. The extra value of depth (or z) is not necessary for rasterization on the viewport. Therefore the projection matrix ($H_{project}$) can be simplified to ($H'_{project}$) as follows

$$H'_{project} = \begin{bmatrix} \frac{2}{r-l} & 0 & -\frac{r+l}{n(r-l)} & 0 \\ 0 & \frac{2}{t-b} & -\frac{t+b}{n(t-b)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{n} & 0 \end{bmatrix}$$

(3.11)

Since the third row and the forth column of the $H'_{project}$ are composed of zeros, they are not necessary for calculation. Since the input for this calculation is the 3D position of the vertices, their position can be expressed using a 1×3 matrix. The result requires less computation since the calculation is simplified to a 3×3 matrix called $H"_{project}$ which is described as

$$H"_{project} = \begin{bmatrix} \frac{2}{r-l} & 0 & -\frac{r+l}{n(r-l)} \\ 0 & \frac{2}{t-b} & -\frac{t+b}{n(t-b)} \\ 0 & 0 & \frac{1}{n} \end{bmatrix}$$

(3.12)

The final rasterized image on the display can be produced from the product of the vertices of objects in the view frustum in the $H"_{project} \cdot H_{trans} \cdot HS_{object}$.

3.2.4. Binocular Vision

Binocular or stereoscopic vision is a term that is used to refer to the perception of depth and 3D, perceived based on receiving visual information, simultaneously from both eyes [55]. Due to the lateral position difference on the head, the eyes receive two slightly different projections from the outside world. This difference is used by the brain to perceive 3D dimensions. The lateral position of the right eye can be found from the cross product of the head sight orientation vector (i.e., *o*-axis or orientation axis) and the head top direction vector (i.e., *a*-axis or approach axis). The cross product of the o-axis and a-axis is known as the *n*-axis or normal axis. The lateral position of the left eye can be found from the negated *n*-axis [56]. Figure 3.5 shows the *a*-axis *o*-axis and *n*-axis with respect to the head.

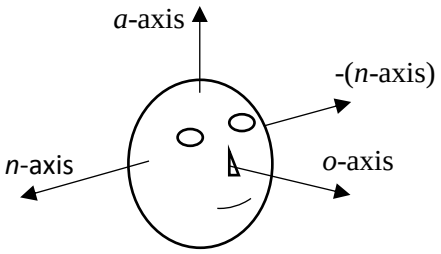


Fig. 3-5. Definition of Orientation, Approach and Normal axis for the head.

The lateral distance between the two scenes (D_L), rendered for the two eyes needs to be determined by the user in the calibration process. A standard model for the head suggests that most people use the distance of 10cm for lateral distance. Therefore, the perspective projection should be applied from each eye independently. Figure 3.6 shows the two overlapped view frustums used for left and right eye perspective projections.

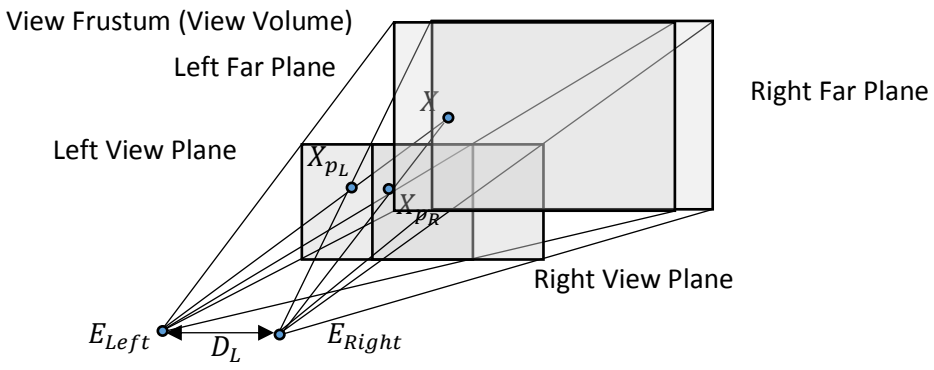


Fig. 3-6. Overlap of left and right view frustum used for stereoscopic rendering.

3.2.5. Head Mounted Display

The head mounted display (HMD), allows the user to freely look around in a VR environment. Unlike a conventional computer's display that only allows the user to see the VR environment from a framed window. An HMD is comprised from High definition liquid crystal display (LCD) display placed in front of two eye pieces (lenses) a head tracking system and additional mechanical and electrical components. Figure 3.7 shows an exploded diagram of the Oculus Rift DK2 head mounted display, used in this study [51].

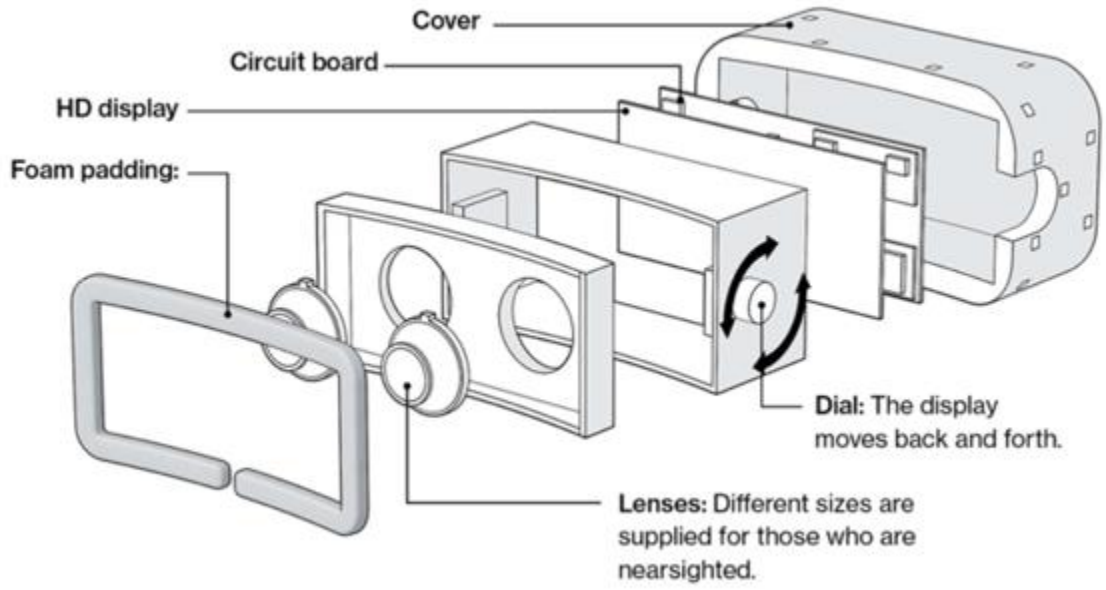


Fig. 3-7. The Oculus Rift DK2 head mounted display [51].

The display shows the two rendered scenes (for the left and the right eye) side by side on the high definition (HD) display. Figure 3.8 shows a sample image displayed on the HD display in the Oculus Rift HMD [51].

Fig. 3-8. Sample image shown on the HD display used in the Oculus Rift HMD [51].

3.2.6. Peripheral Vision

When a scene is placed in front of the two eyes, it will be seen by the central vision or macular vision. In this region, the two eyes can perceive the scene from two different angles and the brain can perceive the sense of depth. The vision outside our macular vision is called the peripheral vision. The peripheral vision area starts from about 18° and stretches to about 110° [57]. Figure 3.9 shows the macular and the peripheral vision regions from a head top view.

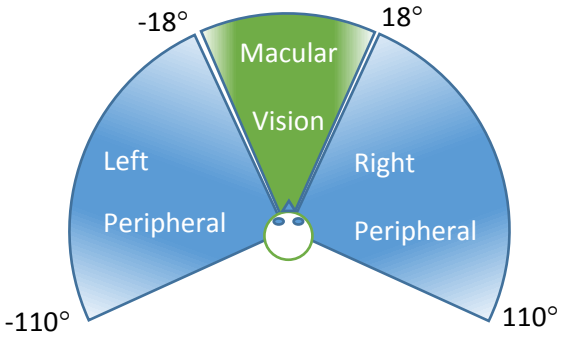


Fig. 3-9. Top view of macular and peripheral vision angles with respect to the head.

An ordinary computer display based VR system can only cover the macular vision due to its limited field of view. On the other hand, HMD based VR systems can cover a higher field of view and therefore more of the peripheral vision area. This will make head mounted display based systems more naturalistic and therefore more immersive. The HMD increases the field of view by using two convex lenses placed in front of each eye. The convex lenses will narrow the focal length and consequently allows the eye to see a wider field of view [58]. Figure 3.10 depicts the effect of placing the convex lens in front of the eye.

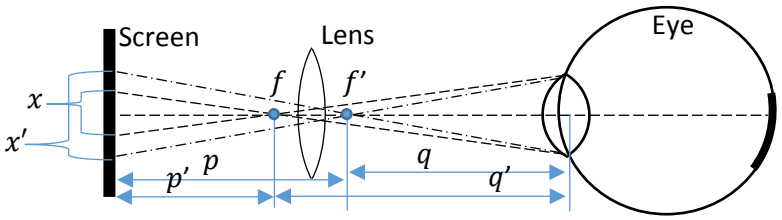


Fig. 3-10.The effect of using a magnifier to increase the view of view of the eye.

The focal length (f) and the narrowed focal length (f') as well as the magnification factor (m) can be determined from

$$\frac{1}{f} = \frac{1}{p} + \frac{1}{q} \tag{3.13}$$

$$\frac{1}{f'} = \frac{1}{p'} + \frac{1}{q'} \tag{3.14}$$

$$m = \frac{x'}{x} = \frac{q'}{q} = \frac{p}{p'} \tag{3.15}$$

The lens used in the HMD increase the field of view with the cost of distorting the image. The distortion caused by the lens is known as pincushion distortion. Pincushion distortion occurs due to the increase of magnification of the lens with the increase of the axial distance. Pincushion distortion caused from the image passing through a lens can be compensated using a synthesized image with barrel distortion on the HMD screen. This will result in a corrected image, perceive by the user. In order to balance the pincushion distortion using barrel distortion, the effects should be of the same magnitude. The magnitude can be defined as a characteristic for both the pincushion and the barrel distortions.

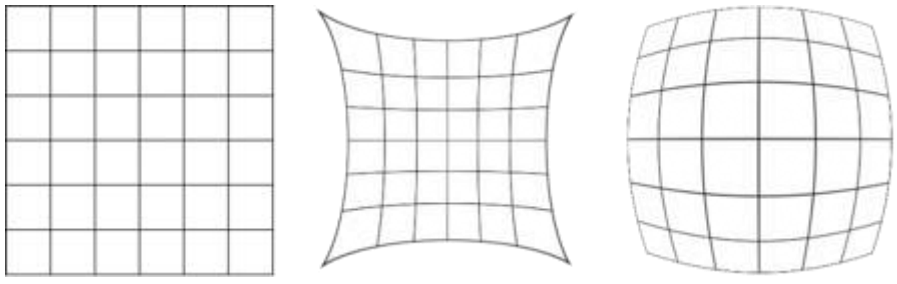
The lens is an axial symmetric object and therefore, it will have radially symmetric distortion. Therefore, using a polar coordinate system is more suitable. However, the pixels in an image are described using their Cartesian coordinates. This leads to using a distortion factor (k) based on (r), which is the distance for any given pixel coordinates (x, y) with respect to the coordinates of the image center $(\bar{x}, \bar{y})$ as given by

$$r = \sqrt{(x - \bar{x})^2 + (y - \bar{y})^2} \tag{3.16}$$

The duplet equations can be described as following

$$\begin{cases} x_{distortion} = x(1 + kr^2) \\ y_{distortion} = y(1 + kr^2) \end{cases} \tag{3.17}$$

where, for $k > 0$ it can lead into the pincushion distortion and for $k < 0$ it results the barrel distortion. Figure 3.11 shows a grid transformed using the pincushion and the barrel distortions.



(a) (b) (c)

Fig. 3-11.A non-distorted grid (a) grid distorted by the pincushion (b) and barrel distortions (c).

The lens used on the HMD will have a specific positive (k) value which can be determined using a calibration process. The calibration process determines the (k) value and the reversed ($-k$) value distortion (barrel distortion) is applied to the image to compensate the lens distortion for the preserver. The other calculation required to apply the perspective projection are based on matrix multiplication ($H"_{project} . H_{trans} . HS_{object}$), while the barrel distortion is a non-linear operation and requires the calculation of the distance from the center of the image (r). On the other hand, the optimization process for the ($H"_{project}$) eliminates the need for calculating the (z) value. However, the (z) value will be recalculated independently. The (z) value and the unit value can be replaced with xr^2 and yr^2. The result can be shown as

$$\begin{bmatrix} x_{distortion} \\ y_{distortion} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & k & 0 \\ 0 & 1 & 0 & k \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ xr^2 \\ yr^2 \end{bmatrix} \quad \textbf{(3.18)}$$

where:

$$r^2 = (x - \bar{x})^2 + (y - \bar{y})^2 \quad \textbf{(3.19)}$$

The following calculations project the vertices from the VR environment world, constructed on the near plane onto to the output screen. The vertices resemble the triangles, which resemble the objects. The triangles should be filled with corresponding pixels where the vertices are not present. The filling pixels are calculated using interpolation by the rendering system. The rendering system considers several values, which are called the surface attributes. The surface attributes determine the overall appearance of each triangle to the perceiver based on the object material defined by the VR environment designer.

3.2.7. Head Tracking System and Implementation of Head Movement

The HMD increases the immersion in the VR environment. It covers the macular and peripheral vision. Using the head tracking system implemented on the (HMD), the user can freely look around and observe the required information from the surrounding scene. Since the HMD acts like a window to the VR world, it is necessary to transform the generated scene position to correspond to the movement taken by the user.

The head tracking system used in most HMD systems in based on an inertial measurement unit (IMU). The IMU system can only provide the absolute angles of the head rotation and is not able to provide position information. On the other hand, a typical head movement (looking left, right, up or down) involves translation as well. Since the IMU sensor cannot provide the translation information, it is important to implement the transformation required for head based on a standard model.

The HMD used in this study (Oculus Rift DK2) in equipped with an IMU sensor, which reports the head positon using a unit-length quaternion representation. The quaternion is described as

$$Q(v,\theta) = (q_1, q_2, q_3, q_4) = \left(\cos\left(\frac{\theta}{2}\right), v_x \sin\left(\frac{\theta}{2}\right), v_v \sin\left(\frac{\theta}{2}\right), v_z \sin\left(\frac{\theta}{2}\right)\right) \quad (3.20)$$

where, $v = (v_x, v_y, v_z)$ is rotation vector of the head and θ is the magnitude of the rotation. The *view frustum perspective model* assumes the positon of the eye E to be stationary in a head rotation, while using the HMD, the eye positon moves during rotation [59]. Figure 3.12 shows the

transformation of the eye position based on horizontal head rotation with the consideration of the point of articulation c as a fixed point and the eye offset h and the horizontal rotation of θ_h.

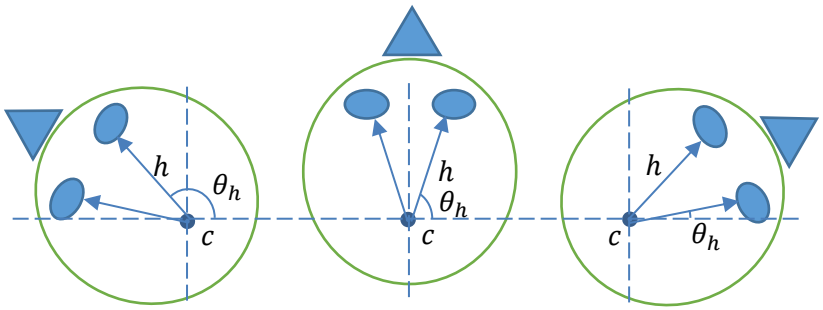


Fig. 3-12.Transformation of the eye positon based on horizontal head rotation.

As it is illustrated in Fig. 3.13 the same principle can be applied to a vertical rotation θ_v of the head with the eye to point of articulation c eye offset of v.

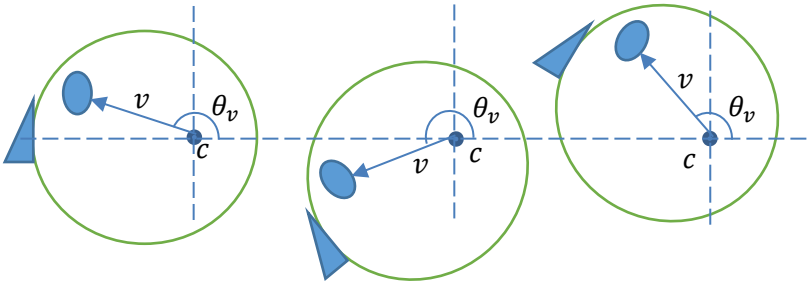


Fig. 3-13.Transformation of the eye positon based on vertical head rotation.

The head tracking values are used to calculate the transformation from point c to eye positions. The transformation matrix can be written as

$$H_{head}(h,\theta_h,v,\theta_v)=\begin{bmatrix} cos(\theta_h)cos(\theta_v) & -sin(\theta_v) & sin(\theta_h)cos(\theta_v) & hsin(\theta_h)cos(\theta_v) \\ cos(\theta_h)sin(\theta_v) & cos(\theta_v) & sin(\theta_h)sin(\theta_v) & vsin(\theta_h)sin(\theta_v) \\ -sin(\theta_h) & 0 & cos(\theta_h) & \frac{h+v}{2}cos(\theta_h) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(3.21)

where θ_v and θ_h are obtained from the quaternion $q(v,\theta)$ received from the IMU sensor using the following equations

$$\theta_v = -\arcsin(2(q_2q_4 - q_1q_3))$$

(3.22)

$$\theta_h = \arctan\left(\frac{2(q_1q_4 + q_2q_3)}{q_1^2 + q_2^2 - q_3^2 - q_4^2}\right)$$

(3.23)

3.2.8. Visual Rendering Optimization

Optimization of rendering can reduce the required processing power and consequently can allow the rendering hardware to increase the refresh rate. Increasing the refresh rate can increase the overall appearance and minimize the latency of VR environment. Clipping and culling are two methods that are used to optimize the rendering load. Objects in the 3D environment are modeled using 3D meshes, where the 3D meshes are constructed from a set of triangles. The culling process determines which triangles are to be drawn and which ones are not to be drawn.

3.2.9. Triangle Normal Vector and Back Face Culling

Every 3D mesh is composed of a set of triangles. Every triangle is created from three vertices, for instance called A, B and C. Vector $\overrightarrow{AB}$ and $\overrightarrow{AC}$ can be created using the three vertices. The cross product of the vectors $\overrightarrow{AB}$ and $\overrightarrow{AC}$ is the normal vector, perpendicular to both. This can be used as the normal vector for a given triangle. Figure 3.14 shows the normal vector obtained from the cross products of $\overrightarrow{AB}$ and $\overrightarrow{AC}$.

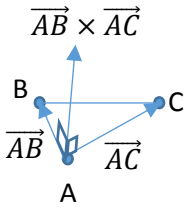


Fig. 3-14. Normal vector obtained from the cross product of two vectors.

The normal vector $\overrightarrow{AB} \times \overrightarrow{AC}$ can be calculated from the matrix multiplication

$$\overrightarrow{AB} \times \overrightarrow{AC} = \begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \\ \overrightarrow{AB_x} & \overrightarrow{AB_y} & \overrightarrow{AB_y} \\ \overrightarrow{AC_x} & \overrightarrow{AC_y} & \overrightarrow{AC_z} \end{bmatrix} \tag{3.24}$$

which results in

$$\overrightarrow{AB} \times \overrightarrow{AC} = \hat{x}\left(\overrightarrow{AB_y}\,\overrightarrow{AC_z} - \overrightarrow{AB_z}\,\overrightarrow{AC_y}\right) - \hat{y}\left(\overrightarrow{AB_x}\,\overrightarrow{AC_z} - \overrightarrow{AB_z}\,\overrightarrow{AC_x}\right) + \hat{z}\left(\overrightarrow{AB_x}\,\overrightarrow{AC_y} - \overrightarrow{AB_y}\,\overrightarrow{AC_x}\right) \tag{3.25}$$

The normal vectors are oriented toward the outside of any given object that they comprise. The normal vectors can be used to determine whether a triangle needs to be drawn. This is done by

evaluating the dot product (d) of the normal vector of a triangle and the eye vector $\overrightarrow{EA}$ (connecting eye position to the arbitrary point A from the triangle) which can be calculated from

$$d = \left(\overrightarrow{AB} \times \overrightarrow{AC}\right).\overrightarrow{EA} \tag{3.26}$$

In case of (d=0), the triangle is perpendicular to the eye sight. If ($d < 0$) it means that the triangle is behind and is not seen by the perceiver. This is called a culled triangle. Only the triangles with $d \geq 0$ need to be drawn on the screen. The purpose of culling is to reduce the amount of calculating needed to project all the objects in a frustum. Calculation of the triangle normal requires additional processing. On the other hand, the triangle normal is constant for solid and static objects. Therefore, it can be calculated beforehand and stored along the three vertices describing a triangle. Figure 3.15 shows the N vertex representing the normal vector. The normal vector is also called the facet normal and is describe using ($\overrightarrow{AN}$).

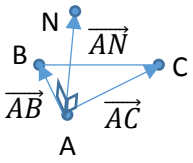


Fig. 3-15. Addition of vertex *N* to eliminate the need for normal vector calculation for a triangle.

By using the pre-calculated normal vector, the calculation of (d) can be simplified to

$$d = \overrightarrow{AN}.\overrightarrow{EA} \tag{3.27}$$

Hence, the triangles with ($d \geq 0$) are called visible constructing triangles and should be drawn while triangles with ($d < 0$) are culled.

3.2.10. Clipping of Boundary Triangles

The process of intersecting the visible constructing tringles of a boundary object in the VR environment with the view frustum planes is to reduce the required rendering-processing load by eliminating unnecessary parts of objects, located partially in the view frustum. In clipping a triangle with a plane, if two vertices of a triangle fall outside the view frustum the result is a new triangle as shown in Fig. 3.16 a and if one of the vertices falls outside the view frustum the result is a quadrilateral as shown in Fig. 3.16.b. In the case of a quadrilateral, the result is split into two triangles.

(a) (b)

Fig. 3-16.Clipping of a triangle can result in either a new triangle (a) or a quadrilateral (b).

To find (N_1) and (N_2) first the vertices of the triangle are evaluated with the view frustum. Then the intersecting vectors are found from connecting an inner vertex with an outer vertex (in this case $\overrightarrow{AC}$ *and* $\overrightarrow{BC}$). The intersecting points can be found by solving the following equations for u_1 and u_2:

$$\begin{cases} N_1 = A + u_1(B - A) \\ N_1 \in K \end{cases} \tag{3.28}$$

$$\begin{cases} N_2 = A + u_2(C - A) \\ N_2 \in K \end{cases} \tag{3.29}$$

3.2.11. Rasterization on the View Plane

The process of turning the triangles into a rasterized surface, ready to be displayed on the user screen, is called rasterization. This process includes defining the filling pixels of visible constructing triangles, theirs colors based on the vertices and the effect of surrounding light sources. A color vector (*R, G and B*) can be assigned to each vertices. The interpolation algorithm can rasterize the filling pixels based on these values. This method is used for relatively simple objects that do not require a complex texture wrapping.

Lighting, Light Sources and Shadows

Whether interpolation or texture wrapping assigns the filling pixels, lighting effects should be applied to induce the effect of surrounding light sources in the VR environment. In addition, an object in the presence of a light source may create a shadow. Replicating the lighting and shadow effect makes the overall appearance of the VR simulation more naturalistic. There are three types of light source in a VR environment. Spotlight, point light and directional light. Lights can have different wavelengths based on the application [53].

In the real world, different materials have different reflection properties such as ambient ($\vec{P}_A$), diffuse ($\vec{P}_D$) and specular ($\vec{P}_S$). Therefore, the total lighting ($\vec{L}_T$) for a pixel that can also emit light ($\vec{L}_E$) is calculated using

$$\vec{L}_T = \vec{L}_E + \vec{L}_A + \vec{L}_D + \vec{L}_S \tag{3.30}$$

In the case of a light passive pixel, $\vec{P}_E$ is considered to be zero.

The lighting is applied by calculating the color vectors for a triangle by considering all the surrounding light sources. There are three methods to calculate lighting, Phong, Gouraud and flat [60]. In the Phong method, lighting calculation is done for all the individual filling pixels in a triangle. It gives the most naturalistic results in terms lighting with the cost of heavy processing load. On the opposite side, the flat method calculates the lighting values for a single pixel in the triangle (usually the first vertex) and applies it the entire pixels. It requires minimum processing load while the result is compromised. In the Gouraud method, the lighting is calculated for all the three constructing vertices of a triangle and the filling pixels are calculated using interpolation. This method demands relatively low processing load while the result is acceptable in most cases.

For every point, the color is stored in a vector ($\vec{P} = [R, G, B]$). Which R, G and B are the normalized Red, Green and Blue values (between 0 and 1).

Ambient Light

Passive objects that are not emitting any light are seen based on the reflection of the ambient light sources. Assuming the light source has a color composition of ($\vec{L} = [R, G, B]$) and intensity

of (I), the outcome color $(\vec{L}_A)$ with the subscript $(_A)$ representing ambient, the ambient light can be calculated using the equation

$$\vec{L}_A = \vec{P}\,(I\,.\,\vec{L})$$

(3.31)

Diffuse Light

The diffuse light reflected from an object resembles a matt or rough surface. The diffuse light can be calculated using the Beer-Lambert-Bouguer Law which describes the intensity of the reflected light by the cosine of the angle between the triangle normal $(\vec{N})$ and the light direction $(\vec{D})$. Since lighting an object from behind is not possible for an opaque object, the angle can only be between $-90°$ and $90°$. The diffuse light $(\vec{L}_D)$ can be found using

$$\vec{L}_D = \vec{P}\,[\vec{N}\,.\,\vec{D}]\,(I\,.\,\vec{L})$$

(3.32)

Specular Light

White diffuse light is the reflection of a matte surface; the specular is the light reflection for a shiny surface. A combination of both can resemble any arbitrary composition of matte and glossy for any given material. In order to calculate the specular light, it is necessary to consider the perceiver direction $(\vec{E})$. The angle between the light direction $(\vec{D})$ and the triangle normal vector $(\vec{N})$ should be within a limited range and is denoted by (L_r) and is calculated using

$$L_r = |[\vec{N}\,.\,\vec{D}] - [\vec{N}.\,\vec{E}]|$$

(3.33)

Thus, the specular light is calculated using

$$\vec{L}_S = \vec{P}\left[\vec{N}\,.\,\vec{D}\right](I\,.\,\vec{L})(1 - L_r^2)$$

(3.34)

A combination from all the different lighting effects can result in any arbitrary composition to resemble certain material for the virtual objects and surfaces in the VR environment.

Shadowing, the Effect of Light Absence

In an appropriate lighting system, every pixel on the view plane is evaluated by the rasterizer. This will result in shadowing effect for surfaces that the light source is blocked from shining directly on. However, in the case of a large flat surface, it is necessary to consider alternatives to consider lighting effects on a part of a triangle rather than the entire triangle.

Far Plane Aliasing, Blurring and Fog Effect

Perspective projection rendering can result in aliasing noise for pixels as they approach the far plane. This is due to the quantization induced by the limited number of pixels rasterized on the screen. Aliasing can cause these pixels to resonate and jitter between different values and ultimately an unpleasant outcome for the observer. In order to eliminate the aliasing problem a combination of blurring and fog effect can be used. This is done by extending the far plane (ef) and projecting the extended volume on the far plane with a smoothing filter (blurring or fog). Figure 3.17 shows the extended far plane with respect to the view frustum [53].

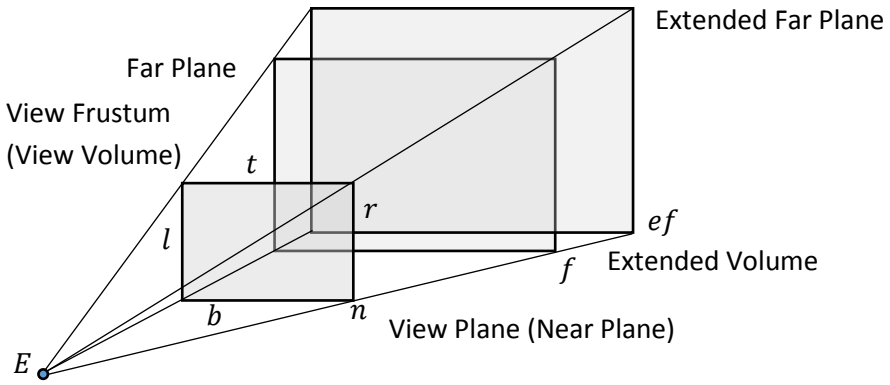


Fig. 3-17.Extended volume and the far plane.

After rendering the far plane using perspective projection, the result should be filtered to eliminate aliasing. The filtering can be done either using blurring or fog or a combination of both. A fog effect is more suitable for an outdoor environment while blurring can be used for both. In blurring, a scrolling moving average is applied, which smooths the contents on the far plane. Adding the fog coefficient (F_c) in the following equation can produce fog effect by using

$$\vec{P}' = \left(F_c \,.\, \vec{L}_F\right) + (1 - F_c)\,\vec{P} \tag{3.35}$$

where, ($\vec{L}_F$) is the color vector of the fog. The fog coefficient changes based on the distance from the perceiver (E) can be applied based on an exponential function.

3.3. Sound Rendering

In the early development of video games and VR environment, the emphasis was only on visual rendering and less development and research was done on the audio rendering. Interestingly, the visual sense is limited by a field of view while the auditory sense is omnidirectional. We perceive our surrounding world using our eyes and our ears. In addition to seeing the world with our eyes, we see the world using our ears by perceiving the sound effects caused by the environment [61].

Since the VR environment tries to simulate physical presence, it is important to render the expected sounds in addition to the visual scene. In order stimulate the auditory sense based of the corresponding sound sources it is necessary to consider the structure and the material assumptions in a VR environment. Similar to stereoscopic rendering that adds the sense of depth to the synthesized image and makes you see through the eyes of a virtual character, binaural synthesis gives the sound the sense of direction and makes you hear through the ears of a virtual character.

3.3.1. Synthesis of Binaural Sound

It might have been an evolutionary reason why we are capable to localize an audio source based on what we perceive using our two ears [62]. The collaboration between the brain, the inner ear and the outer ear (pinna) allows us to figure out the location of an audio source by comparing what the two ears hear based on the lag between similar patterns. The Doppler Effect is used to deduce if an audio source is approaching or is departing. Additionally, we can sense the surrounding structure as well as its composition material from the reflections received by the ear.

The audio rendering starts from triggering the synthesizer. A sound source can be represented an object or the background noise related to an environment. Based on the application and level of details, an audio source can be enabled or disabled. By enabling every audio source, an individual sound-rendering pipeline should be considered. Figure 3.18 shows an individual sound rendering pipeline.

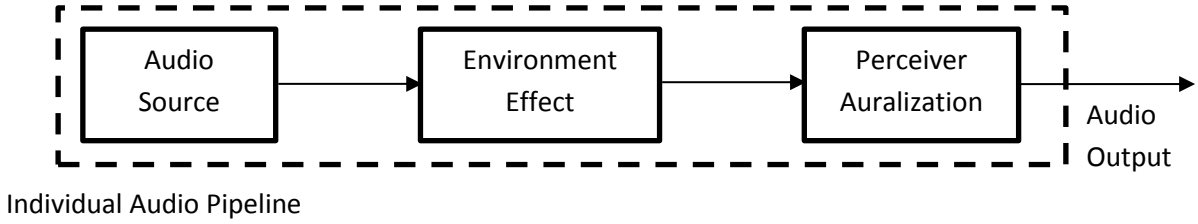


Fig. 3-18.Individual audio pipeline for an object in the virtual environment.

The environment effect includes the frequency shift, attenuation and echo effect caused by the environment on the audio source. The perceiver auralization can be performed either using two speakers (such as headphones) or multiple surrounding speakers (such as with surround sound 5.1 or 7.1 systems). The multiple surrounding speaker method can deliver the sense of direction for audio with the cost of multiple speakers. This method requires an anechoic environment and the speakers need to be placed stationary and calibrated based on the position of the perceiver. The perceiver should remain stationary with respect to the speakers. In contrast to this method, the two-speaker method can be implemented using ordinary headphones. In combination with a head mounted display and the head tracking sensor, it is possible to deliver the sense of direction without the need for keeping the perceiver stationary.

The two-speaker method is based on the calculation of the sound field for the two ears using the head related transfer functions (HRTF) [63]. The HRTF defines the corresponding phase shift

and attenuation for each ear based on the location and the attitude of the perceiver relative to the audio source. The HRTF may vary for different people based on their head and the outer ear size and shape. However, a standard model (average model) can be used to avoid the need for calibration of every user. As shown in Fig. 3.19 the outcome from all individual audio sources goes a mixer and is played back for the user.

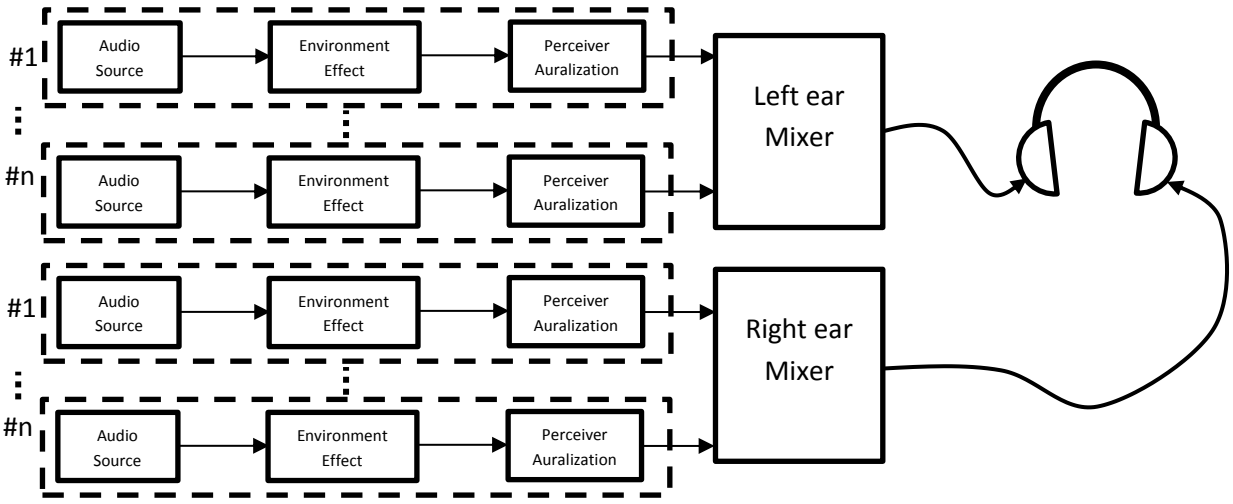


Fig. 3-19.Left and right ear mixer adding the individual audio pipeline.

3.3.2. Echo and Reverb Effects

Echo and reverb can imply a spatial perception, perceived from hearing. Sound waves travel in all directions form an audio source. When sound waves hit a surface, based on the composition of the surface, parts of it will be reflected and parts of it absorbed. Depending on the material, the ratio between reflection and absorption can vary. If a person is near an object with considerable sound reflection properties, the reflection can be heard with the delay caused by the speed of sound traveling in air [64].

The delay value indicates the distance to that object. Assuming a person placed in a small room with reflective material used for its walls and is in the presence of an audio source, an echo is perceived. An echo is the reflection from a surface with a distance of more than 17 meters. If the surface is closer than 17 meters, the reflection is called reverb. This is because sound travels approximately 340 meters per second. It takes 100 milliseconds for an audio wave to travel the distance of 17 meters forth and back. While speaking out every single vowel takes minimum of 100 milliseconds, the reflection of the sounds can be received and it is called reverb. While if the distance is higher than 17 meters, it will take longer than 100 milliseconds for the reflection to be heard and it will appear as echo. Therefore, in principle echo and reverb are similar in terms of both are reflection of the sound perceived along with the original sound. This can be utilized in VR to imply the presence of objects in a specific space.

3.3.3. Doppler Shift and Moving Objects

The Doppler shift is the change in the frequency of a wave due to the relative movement of the observer. In the case of an observer moving toward an object, the emitted sound will be perceived higher in frequency, while if the observer moves away from the sound source the frequency is perceiver lower. This phenomenon is used to give the impression of approaching or departing from an audio source in the VR [65]. The shift in the perceived wave frequency (Δf) is proportional to the relative velocity (v_r) and is found using

$$\Delta f = \frac{v_r}{v_s} f_0 \tag{3.36}$$

where, (v_s) is the velocity of sound traveling in air and (f_0) is the original wave frequency.

3.4. Physics Engine and Physical Interaction

A physics engine is designed to simulate a limited selection of classic mechanics for large objects (mainly rigid) under the influence of gravity and other forces. Since VR tries to mimic the real world, it is necessary to mimic phenomena such as collisions (rigid body dynamics), inertia, elasticity and gravity, as accurately possible in the simulation [66]. The physics simulation can be performed with either precision or performance as priority.

In the case of scientific simulations or animated movies, precision is prioritized. In some cases, physics simulations require super computers and the result cannot be expressed in real time as they have a long runtime requirements. On the other hand, in video games and VR platforms, physics simulation should be performed in real time to give the user the illusion of immersion and therefore performance is prioritized. If the physics is not implemented properly, objects will move and behave in odd manner. This can reduce the immersion for the perceiver and therefore, the physics engine should make the interaction and the movement of objects in the VR environment to look as real as possible [67].

It is vital to implement the three elements in the physics engine to simulate the physics of objects interacting with forces similar to their behavior in reality. Every real system in mechanics can be described using a three-element system comprised of spring-damper-mass interacting with induced forces. Figure 3.20 shows a model of a cart placed on a rail, with the mass (m), attached to a sprint with the spring-constant (k) and a damper with the damping factor (b) interacting with the force (F).

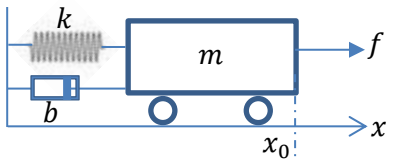


Fig. 3-20.Sprint-Damp-Mass model of a cart.

3.4.1. The D'Alembert's Principle and the Interaction between Forces

The D'Alembert's principle states that the sum of all the forces applied to an object are equal to zero. For instance, an object resting on a surface is facing gravity while the surface applies enough force to counter balance the object to stay on the surface in a stationary position. The mathematical sum of the forces (f_i) can be shown using

$$\sum_i f_i = 0 \tag{3.37}$$

Based on the second Newton's law of motion, if a motive force (f_{motive}) is induced to an object with mass (m) it will cause it a proportional acceleration (a). As time passes, the acceleration builds up velocity (v). The increase of velocity increases momentum (p) proportional to the objects mass. Assuming the object moves in an environment with the presence of air, by the increase of velocity, kinetic friction (f_{Kfric}) and drag (f_{drag}) increase proportionally [68].

$$f_{motive} = ma + f_{Kfric} + f_{drag} \tag{3.38}$$

$$f_{Kfric} = \mu_k mg \tag{3.39}$$

$$f_{drag} = \frac{1}{2}\rho A v^2 \mu_D \tag{3.40}$$

where μ_k is the kinetic friction coefficient ρ represents the density of air, $\mu_D$is the drag coefficient, A is the object cross sectional area perpendicular to the direction of movement with the velocity of v. In a more accurate model the static friction (f_{Sfric}) can be considered with the cost of more processing power.

3.4.2. The Hook's Law and the Spring Effect

Hook's law describes the spring effect in a given object. The spring effect can describe the deformation (particularly compression (Δx)) of an object due to an applied force (f_s). The deformation is proportional to the force and the (k) spring-constant, defined for that object for a specific coordinate and can be written as

$$f_s = -k\Delta x \tag{3.41}$$

The excretion of a force on a sprint will compress it proportionally. Once the force is taken away, the spring retracts to its original size and might oscillate around it. The simulation of the spring can be expanded to simulate the behavior of cloth, flags, ropes or even water ripples [69].

3.4.3. Collision Detection and Collision Response

Based on Pauli's exclusion principle, two identical fermions cannot occupy the same quantum state, simultaneously [70]. In a day-to-day life, a person walks into a wall instead of walking through a wall. In reality, two solid objects cannot pass through each other, while in VR solid objects are able to pass through each other since there are only a simulation. To make objects in VR interact the same way as they do in reality, a collision detection and collision response is

required. The collision detection finds the intersection between objects and reports it to the collision response. Collision response simulates the collision physics. Collision physics is based on mechanical principles of motion such as impact, forces, elasticity, gravity and friction.

As mentioned previously, a person (a camera), navigating in a VR environment should not be able to walk through a wall. This can be done by confining the position of the person to the surrounding walls and obstacles. In case of placing a person in a virtual room, the navigation control commands would be, blocked or executed based on the person's position versus the room walls.

The following method can be used to prevent the VR perceiver to walk through a wall. This method can be expanded to other rigid obstacles in the VR environment where the person should not be able to walk through them. This is done using the collision detection while the collision reaction only blocks the navigation commands. The collision detection considers the wall as a plane and the person can be represented using a bounding volume like a bounding sphere or a bounding box. The bounding volume should be tight as possible to provide enough accuracy for the collision detection. The navigation input commands are executed unless there is an intersection found between the plane and the sphere. This method can be expanded to other objects in the VR environment.

Evaluation of collision between all the objects in a large virtual reality environment requires high processing resources. Additionally, evaluation of dynamic objects is more complex than static objects since the dimension of time should be considered as well. This makes the collision detection an extensive task. On the other hand, evaluation of collision for each object with all other objects in the virtual world is not necessary since objects can only collide with others objects that

are in their vicinity. One way to optimize the collision detection process is to organize objects based on their position, in collision groups. A bounding volume hierarchy (BVH) can be utilized for this application [71].

After collision detection between two objects, it is necessary to report the collision point to the collision response. Collision response uses the point of collision point and the position of the two objects prior to the collision to calculate the corresponding physics associated with the collision. The simulated physics can be as simple as pushing the two objects apart to the state where collision is avoided toward the opposite angle of impact. The angle of impact is obtained from the movement vector and the surface normal vector. This will simulate the bouncing effect of an object off another object. Moving the objects apart will be executed with a velocity, elasticity, restitution, friction and the effect of gravity based on the status and the attributes of the colliding and the neighbor objects.

3.4.4. Inertia, Elasticity and Gravity

Every object in the VR environment is described using a state variable (location (X_i), orientation (θ_i), linear velocity (v_i), linear acceleration (a_i), angular velocity (ω_i) and angular acceleration (α_i)) shown as

$$S_{obj-i}(X_i, \theta_i, v_i, a_i, \omega_i, \alpha_i) \tag{3.42}$$

In case of collision between two objects, the kinetic energy in the objects is converted into potential energy, causing objects to compress (even slightly) and creating heat. Then the objects try to return

to their normal state by releasing the store potential energy into kinetic energy, causing the two collided objects to separate from the collision point. The comparison between the relative velocity (V_r) of the objects, before ($t < t_c$) and after ($t > t_c$) the collision ($t = t_c$) is called the coefficient of restitution (C_R) and can be calculated using

$$C_R = \frac{V_r(t > t_c)}{V_r(t < t_c)} \tag{3.43}$$

The (C_R) varies based on the elasticity, shape and material of the objects engaged in the collision. Based on the location of an object and the interaction with the surrounding objects (linear forces, angular forces and physics laws), the state variables change. Gravity affects all the objects by applying a linear force proportional to their designated mass in the opposite direction of the global *y*-axis. This force leads into acceleration ($9.8\frac{m}{s^2}$) for simulations assuming physical presence on the surface of the earth.

3.5. Intuitive Input Interface

The VR platform should be able to establish a dialog with the user. This dialog should be similar to what a user experiences with the real world. One key element in establishing such a dialog is the intuitive interface both from input and output side. The VR environment was initially designed based on an ordinary input device like a joystick. A joystick requires computer skills and might cause difficulties in usage for an inexperienced user. This difficulty may cause a bias in the result due to computer experience rather than spatial navigation performance. To solve the problem of

computer usage bias in the data a custom hardware solution called VRNChair was developed which is explained further [72].

3.5.1. The design of the VRNChair

There are different types of wheelchairs. However, the most popular ones consist of two large main wheels attached to either side of the center of the wheelchair and two additional smaller wheels or casters on both sides of the front of the wheelchair as shown in Fig. 3.21. For this application the Drive Medical, Silver Sport 2 – Dual Axle wheelchair was selected. A custom-made motion capture unit developed and attached to the wheelchair.

Fig. 3-21.VRNChair, based on an ordinary wheelchair.

The velocity of the two main wheels (left and right) causes the wheelchair to have a combination of translational and rotational moves. The participant sits on the wheelchair while a laptop is placed on a wooden tray resting on the wheelchair handlebars. Then, the participant navigates through the VR environment by moving the wheelchair as the input device. The motion capture unit senses the movement of the wheelchair and feeds the VR platform. Experimentation showed that using this method as opposed to using an ordinary joystick comes with two main benefits. Firstly, it provides an intuitive way (it does not require learning or skill to use) to give navigation input commands to the VR platform; and secondly, it reduces the effect of the motion sickness (by making the user actually move in accordance to what they perceive from the VR, concurrently).

The VRNChair can move in two different ways, and can combine both: translation ($V_{Up,Down}$) and rotation ($V_{Left,Right}$). The magnitude of velocity for each wheel is given using two vectors, (V_L) and (V_R). The velocity vectors of the two wheels induce two types of motions on the center of the wheelchair, ($V_{Up,Down}$) and ($V_{Left,Right}$). The difference between the velocity magnitudes of the two wheels causes rotational velocity. In addition, the regularity of the radii of the two wheels causes translational movement. Based on the described relationship between the velocity of each wheel and the motion of the wheelchair, a kinematic model was developed in which the ($V_{Up,Down}$) and ($V_{Left,Right}$) are calculated from the (V_L) and (V_R). Therefore, when the sum of the magnitude of (V_L) and (V_R) turn on the same direction it causes the wheelchair to move forward or backward which is the translation component. In contrast, the difference between (V_L) and (V_R) denotes the rotation of the wheelchair [72].

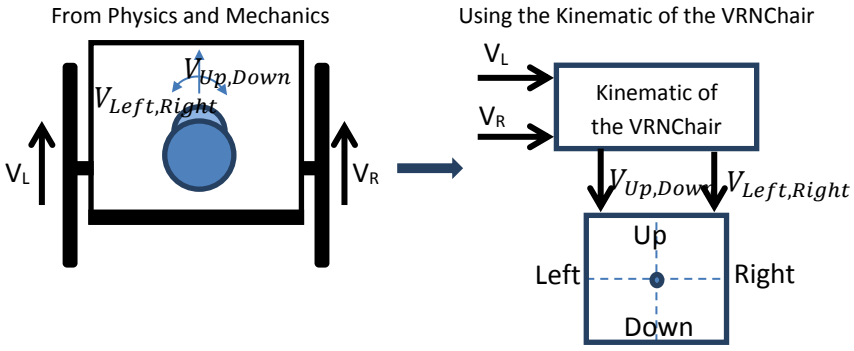
$$V_{Up,Down} = \frac{V_L + V_R}{2}$$

(3.44)

$$V_{Left,Right} = V_L - V_R$$

(3.45)

The outcome of the kinematic model of the wheelchair shown in Fig 3.22 and Fig 3.23. was used as the input device and consequently for the VR environment. Based on the kinematic model and the velocity magnitude acquired by the encoders of both main wheels, the motion capture unit estimates the overall movement of the VRNChair. The estimated motion of the VRNChair passes through a low-pass filter to assure the quality of the estimation and reduction of any unwanted noise [72].



(a) (b)

Fig. 3-22.Head directions vs the wheelchair (a), Kinematic model of the VRNChair (b).

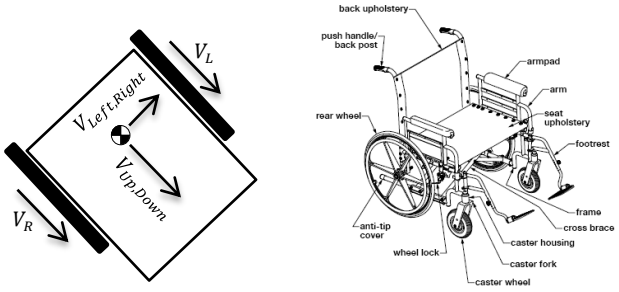


Fig. 3-23.Velocity vectors (a), Schematic of the wheelchair used for the VRNChair (b) [73].

The participant moves the VRNChair by applying force to the wheels. In the case of using the VRNChair as a walker, they move the VRNChair by pushing the handles. Due to the friction between the VRNChair and the ground, motion applied to the VRNChair will cause relative rotation by the wheels. The relation between motions of the wheels caused by the motion of the VRNChair obeys the inverse kinematic model. Estimation of the real motion of the VRNChair is the main function of the implemented motion capture unit. As a result, the unit simulates a standard joystick on the host computer, which receives the resulting motion estimation as a velocity vector. Figure 3.24 depicts the motion capture unit and the human interface device (HID) control panel (Game Controllers) on Microsoft Windows®.

(a) (b)

Fig. 3-24.VRNChair controller motion capture (a), Microsoft® Game Controller for VRNChair (b).

The VR experiment should be calibrated such that distance traversed in the physical environment by the VRNChair is reflected by a similar distance in the virtual environment. Rotation, however, should be calibrated such that a rotation of 360˚ by the VRNChair in the physical environment produces exactly 360˚ of rotation in the virtual environment. All experiments should be calibrated in the same way to maintain consistency among results.

3.6. Artificial intelligence and data analysis

The VR platform is designed to support spatial cognition studies. The VR environment can be a static setup or a hybrid including dynamic objects that are placed based on a sequence or a random order upon a trial. The VR platform can assess the environment for the shortest distance required to reach a target. This evaluation can be used for comparison with a pathway taken by a participant. The difference between the two pathways can be used to benchmark the participant's performance.

Artificial intelligence methods can be utilized to automate the process of finding the shortest distance from a given position to a target position. There are many algorithms designed to find the shortest distance between two given points with the presence of obstacles and physical limitations [74]. In the VR environment, the physical limitations can be as the presence of gravity and not being able to walk through obstacles and walls.

The path-finding algorithm should mimic a human like behavior in solving the navigation task. For instance, in order to approach a window inside a building, it is necessary to walk toward a door, then pass through the door and approach the window from inside the building. Therefore, certain checkpoints are considered which all together form a graph. The graph is presented as an optimization problem. Every path between nodes is rated based on its distance. Using Dijkstra algorithm the shortest path is found and the result is presented as the minimum distance to reach the target [75].

For instance, a floor plan can be represented a graph with different nodes representing inevitable checkpoints that will be visited if the participant decides to go through that specific pathway. Dijkstra algorithm is used to find the shortest distance in order to provide minimum distance

required to reach a target. The minimum distance required to reach a target is used to benchmark a participant's performance. Figure 3.25 shows a node graph representation of the *Virtual House* first floor plan taking the participant to the staircase inside the building. Using the Dijkstra algorithm the path (ABFEJS) with the distance of 16 units was found to be the shortest.

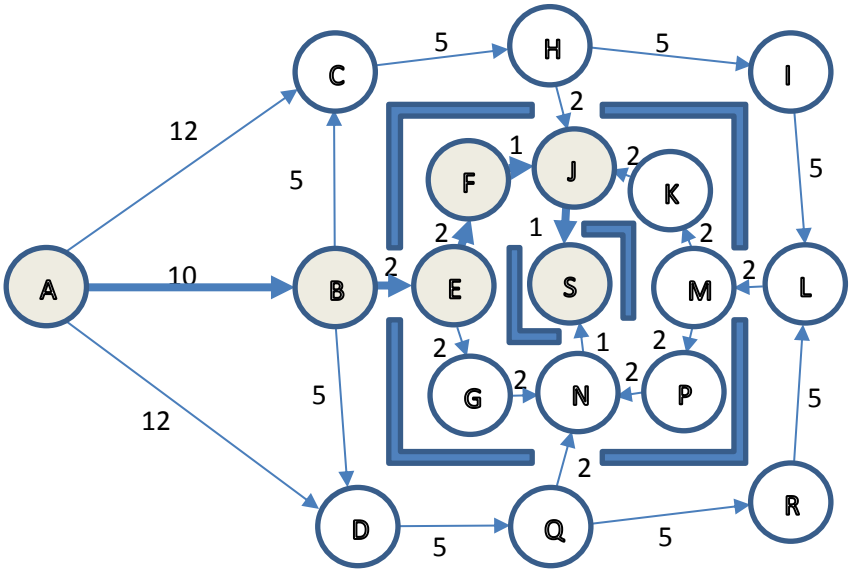


Fig. 3-25.Node graph representing the shortest pathway from a floor plan.

The VR environment engine determines the orientation of the stairs as well as the target window upon the start of the experiment based on a random selection. The nodes are placed upon the placement of the world in specific locations where passing through them is inevitable if the minimum distance is the objective. Subsequently, the minimum distance required to reach the target window needed to be calculated using the Dijkstra algorithm upon the start of the simulation and stored on the log file for further analysis.

3.7. Analysis of the Data

Experimental neuroscience studies are based on analysis of data collected during the experiment. The data includes input commands, elapsed time and the trajectories of movement in the VR environment. One of the advantages in experiments in VR environments is the more accurate, objective and automatic nature of the data collection. In fact, data collection can be part of the game engine and different parameters form the participant as well as the VR environment.

In a spatial navigational experiment, it is necessary to analyze the traversed trajectory of the participants during the desired task. The trajectory is used for comparison, benchmark and further analysis. The metrics include the difference of the traversed trajectory with a reference (optimum) trajectory, the length of the trajectory, traversing speed and locations visited by the participant. Acquiring the trajectory also captures the plausible unintentional extra movements of the participant (i.e. jerky movement of the wheelchair) or the inaccuracy of the positioning sensors; these are considered as position measurement noise. The position measurement noise can negatively impact the accuracy of the comparison and further, the experiment results. Therefore, it is necessary to reduce the unwanted noise contamination using filtering.

As it is shown in Fig 3.1 in section 3.1 the game engine is based on a setup and loop function. The loop function is executed periodically based on the refresh rate. During the execution of the loop function, different parameters of the participant are measured and stored on a log file. The information stored on the log file will be used to analyze the participant's performance.

3.7.1. Spatial and Temporal Equidistance Data

The data collection is based on constant time intervals called the refresh rate. In the case of capturing a trajectory of movement, the samples are temporally equidistant. In other words, the distance between every consecutive data points vary based on the participant's speed. On the other hand, digital spatial filtering requires spatial equidistance data. Therefore, it is necessary to resample the trajectory, spatially.

3.7.2. Spatial Trajectory Resampling

The proposed resampling algorithm is capable to simultaneously suppress or linear interpolate a low-resolution trajectory. The algorithm takes a temporal equidistance set of data with a varying spatial distance and applying up sampling and down sampling based on the demand. This method will provide the best linear estimation of data, which will result in a spatially equidistant data set. The algorithm starts with considering a minimum distance of interest (MDI) shown with the symbol (r) which is used to resample the trajectory. The transformation takes in the data set $x[n]$ and returns the estimated data set $\hat{x}[x]$ as described

$$x[n] = \{p_1, p_2, p_3, \dots p_k, \dots p_n\} \tag{3.46}$$

$$\hat{x}[n] = \{\hat{p}_1, \hat{p}_2, \hat{p}_3, \dots \hat{p}_m\} \tag{3.47}$$

The first data point is passed as the first estimated data point shown with

$$\hat{p}_1 = p_1$$

(3.48)

The algorithm is best described geometrically by considering a sphere with a radius of the minimum distance (r) around the first data point (p_1). The next data point from (p_2) will be compared versus the sphere. That is the distance from the first data point (d_{i-i}) will be used to evaluate the next data point (p_i) and can be written as

$$d_{1-i} = |\, p_1 - p_i| = \sqrt{(x_{p_1} - x_{p_i})^2 + (y_{p_1} - y_{p_i})^2 + (z_{p_1} - z_{p_i})^2}$$

(3.49)

$$k \geq i > 1$$

(3.50)

If the next data point is inside the sphere, it will be skipped until the first data point that is outside the sphere is achieved. Therefore, it can be written as

$$d_{1-i} \geq r$$

(3.51)

Hence, the data point will be marked (p_k)

$$p_k \leftarrow p_i$$

(3.52)

A line is considered between the data point (p_k) and data point (p_{k-1}) which can be written as

$$p_k p_{k-1}(x, y, z) = \left(x_{p_{k-1}}, y_{p_{k-1}}, z_{p_{k-1}}\right) + t(a, b, c)$$

(3.53)

where t is the connecting line and a, b, c are determined by

$$\frac{x_{p_k} - x_{p_{k-1}}}{a} = \frac{y_{p_k} - y_{p_{k-1}}}{b} = \frac{z_{p_k} - z_{p_{k-1}}}{c}$$

(3.54)

The intersection point between the line and the sphere will be considered as the second estimated data point ($\hat{p}_2$). At this point, the sphere and line will be derived by the following equations

Sphere: $||X - p_1||^2 = r^2$

(3.55)

Line: $X = p_{k-1} + \underline{t}e$

(3.56)

where X stand for any arbitrary point in 3 dimensions and e is an extender for the line starting from p_{k-1} crossing p_k. Combining the equation for the sphere and the line, the following equation can be written as

$$||p_{k-1} + \underline{t}e + p_1||^2 = r^2$$

(3.57)

$$(p_{k-1} + \underline{t}e - p_1).(p_{k-1} + \underline{t}e - p_1) = r^2$$

(3.58)

$$e^2(\underline{t}.\underline{t}) + 2e(\underline{t}(p_{k-1} - p_1)) + (p_{k-1} - p_1).(p_{k-1} - p_1) - r^2 = 0$$

(3.59)

which in case of the point after p_1 falling out of the circle (interpolation), the equation will be simplified as

$$e^2(\underline{t}.\underline{t}) - r^2 = 0$$

(3.60)

The result can be calculated from the following delta function

$$\Delta = (\underline{t}.(p_{k-1} - p_1))^2 - ||p_{k-1} - p_1||^2 + r^2$$

(3.61)

Since the line is drawn from two points, one inside the sphere and one outside the sphere, there will be two intersection points between the sphere and the line. This will result in a positive value for the Δ function and equations can be rearranged as follows for a solution of a quadratic equation written as

$$f = (x_{p_k} - x_{p_{k-1}})^2 + (y_{p_k} - y_{p_{k-1}})^2 + (z_{p_k} - z_{p_{k-1}})^2 \tag{3.62}$$

The solution can be written as

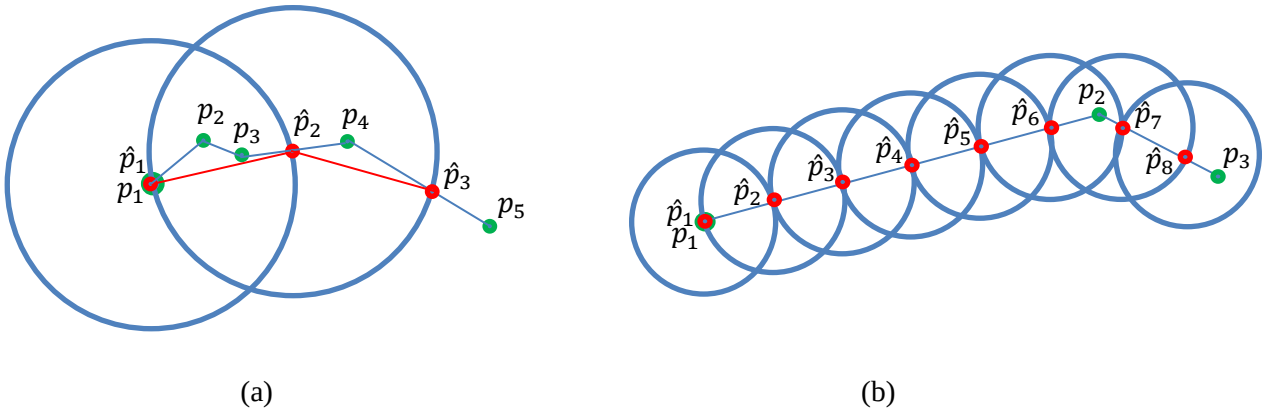
$$e = \frac{-g \pm \sqrt{g^2 - 4fh}}{2f} \tag{3.63}$$

Where (g) and (h) are calculated using

$$g = 2\left[(x_{p_k} - x_{p_{k-1}})(x_{p_k} - x_{p_1}) + (y_{p_k} - y_{p_{k-1}})(y_{p_k} - y_{p_1}) + (z_{p_k} - z_{p_{k-1}})(z_{p_k} - z_{p_1})\right] \tag{3.64}$$

$$h = {x_{p_1}}^2 + {y_{p_1}}^2 + {z_{p_1}}^2 + {x_{p_{k-1}}}^2 + {y_{p_{k-1}}}^2 + {z_{p_{k-1}}}^2 - 2(x_{p_1}x_{p_{k-1}} + y_{p_1}y_{p_{k-1}} + z_{p_1}z_{p_{k-1}}) - r^2 \tag{3.65}$$

The solution will lead to a point between (p_{k-1}) and (p_k) and will be considered the second estimated data point ($\hat{p}_2$). The algorithm iterates by considering a sphere with the radius of (r) around ($\hat{p}_2$). The algorithm keeps iterating until it achieves the last data point. Since the estimated data points are within r distance apart from the neighbor data, they are spatially equidistant. A graphical representation of the resampling is shown in Fig. 3.26.



(a) (b)

Fig. 3-26.Compression or down sampling (a) and interpolation or up sampling (b).

3.7.3. Digital Spatial Filtering

Since the trajectory data is resampled spatially, filtering can be as trivial as scrolling a simple or a weighted moving average on the data. The moving average will have a spatial low pass filter effect and based on the Nyquist frequency, the length of the moving average and the MDI (acting as the sampling rate) will define the cut-off spatial frequency. Considering the resampled trajectory

($x[n]$) and the output of the moving average ($y[n]$) with the length of (N), the calculation can be performed as

$$y[i] = \frac{1}{N}\sum_{i}^{i-N} x[i] \tag{3.66}$$

In the case of the simple moving average, the spatial frequency response can be calculated based on the Fourier transform of a rectangular window. The frequency response based on the relative frequency ($H(\omega)$) for a moving average with length (N) can be written as

$$H(\omega) = \frac{1}{N}\sum_{i-N}^{i} e^{-j\omega i} \tag{3.67}$$

This can be expressed as the sum of the first N terms of geometric series and can be written as

$$H(\omega) = \frac{1}{N}(e^{-j\omega(i-N)} + e^{-j\omega(i-N+1)} + e^{-j\omega(i-N+2)} + \cdots + e^{-j\omega i}) \tag{3.68}$$

By multiplying both sides of the equation by the term $(1 - e^{j\omega})$

$$H(\omega)(1 - e^{j\omega}) = \frac{1}{N}(1 - e^{j\omega})(e^{-j\omega(i-N)} + e^{-j\omega(i-N+1)} + e^{-j\omega(i-N+2)} + \cdots + e^{-j\omega i}) \tag{3.69}$$

Which can be expanded to

$$H(\omega)\left(1-e^{j\omega}\right)=\frac{1}{N}(e^{-j\omega(i-N)}+e^{-j\omega(i-N+1)}+e^{-j\omega(i-N+2)}+\cdots+e^{-j\omega i}-e^{-j\omega(i-N+1)} \\ -e^{-j\omega(i-N+2)}-\cdots-e^{-j\omega(i-1)})$$

(3.70)

This can be simplified to

$$H(\omega)\left(1-e^{j\omega}\right)=\frac{1}{N}\left(1-e^{-j\omega N}\right)$$

(3.71)

Thus, it can be concluded as

$$H(\omega)=\frac{1}{N}(\frac{1-e^{-j\omega N}}{1-e^{j\omega}})$$

(3.72)

Where the relative frequency (ω) can be converted into spatial frequency using the MDI (r) and the equation

$$\omega=2\,\pi\,r$$

(3.73)

The spatial cut-off frequency is written as

$$H(r)=\frac{1}{2\pi N}(\frac{1-e^{-2j\pi rN}}{1-e^{2j\pi r}})$$

(3.74)

Where (r) is minimum distance defined by the resampling and (N) is the length of the simple moving average filter.

3.7.4. Comparison of Trajectories

The filtered trajectory is used for further comparison. Two methods are used for the comparison: one a scalar and the second a vector mode. In the scalar mode, the length of the filtered trajectory is compare with the minimum distance required to reach the target obtained from the Dijkstra algorithm. The vector method uses the filtered trajectory and the shortest distance trajectory found by the Dijkstra algorithm and calculates the surface area between the two trajectories. Figure 2.27 shows an example of the surface area between the two trajectories. In order to calculate the surface area, the two trajectories (J and K) should be resampled spatially using a similar MDI value. Since the two trajectories are spatially equally sampled, the surface area between them can be calculated using the formation of triangles.

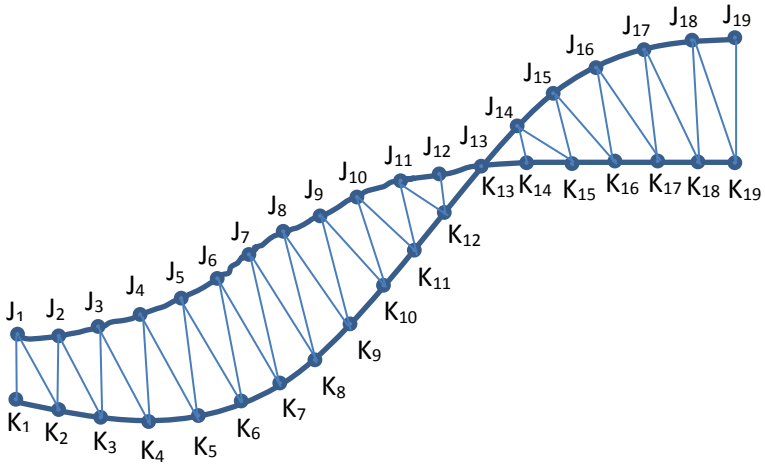


Fig. 3-27.Triangle formation used to calculate the surface area between two trajectories.

The surface area between the two trajectories ($A_{J,K}$) is the sum of the triangles formed between the two trajectories and can be written as

$$A_{J,K} = \sum_{m=1}^{M} S(K_m, J_m, K_{m+1}) + \sum_{n=1}^{N} S(J_n, K_{n+1}, J_{n+1})$$

(3.75)

Where the function $S(p_1, p_2, p_3)$ calculates the surface area of a triangle formed between the points (p_1), (p_2) and (p_3). The three points can form two vectors $(\overrightarrow{p_1p_2})$ and $(\overrightarrow{p_1p_3})$. The surface area of the triangle formed from the three points equals

$$S(p_1, p_2, p_3) = \frac{1}{2}\det(\overrightarrow{p_1p_2} \times \overrightarrow{p_1p_3})$$

(3.76)

Every point is considered in 3D and is represented by three values. For instance, (p_1) is represented by the set of (x_{p1}, y_{p1}, z_{p1}). Thus, the function $S(p_1, p_2, p_3)$ can be written as

$$S(p_1, p_2, p_3) = \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ x_{p2} - x_{p1} & y_{p2} - y_{p1} & z_{p2} - z_{p1} \\ x_{p3} - x_{p1} & y_{p3} - y_{p1} & z_{p3} - z_{p1} \end{vmatrix}$$

(3.77)

Therefore, the surface area between the two trajectories J and K is calculated by

$$A_{J,K} = \frac{1}{2}\sum_{m=1}^{M}\begin{vmatrix} 1 & 1 & 1 \\ x_{Jm} - x_{Km} & y_{Jm} - y_{Km} & z_{Jm} - z_{Km} \\ x_{Km+1} - x_{Km} & y_{Km+1} - y_{Km} & z_{Km+1} - z_{Km} \end{vmatrix} + \frac{1}{2}\sum_{n=1}^{N}\begin{vmatrix} 1 & 1 & 1 \\ x_{Kn+1} - x_{Jn} & y_{Kn+1} - y_{Jn} & z_{Kn+1} - z_{Jn} \\ x_{Jn+1} - x_{Jn} & y_{Jn+1} - y_{Jn} & z_{Jn+1} - z_{Jn} \end{vmatrix}$$

(3.78)

The scalar method measures the length of the traversed distance with the minimum length required to reach the target. This method cannot discriminate between two different trajectories with the same length. In order to consider the shape of a trajectory, the vector-based method can be used. Based on the requirements of an experiment, one of the two methods can be utilized.

3.8. Summary

In this chapter, the theory behind the implementation of a game engine for the VR platform was presented. The visual and auditory rendering systems were described in detail. To use the VR platform for cognitive studies experiments, data collection and analysis is required. The data analysis and benchmarking method was presented. The benchmarking is based on the comparison of the trajectory obtained from a participant, performing the test versus an optimal trajectory calculated by the game engine. A novel method for resampling and filtering of the obtained trajectory was presented to improve the accuracy for benchmarking.

Chapter 4

Design and Implementation of Experiments

Studying navigational spatial cognition requires a physical test environment. Using a physical test environment can become challenging in terms of cost, flexibility, modifiability and less ideal in terms of consistency. A virtual reality (VR) environment can be used as an alternative if the patient performs similar to a test in a real environment. That is, the VR based experiments should map similar results to an experiment in the real environment.

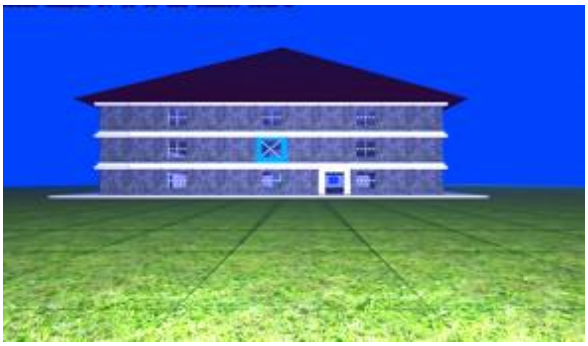
In this dissertation, the objective is to build a landmark-less VR environment to assess spatial cognition and in particular spatial updating that occurs in landmark-less environments. Most commonly in a landmark-less environment, people use the egocentric (self-to-object) spatial updating to encode the surrounding environment based on a first person perspective and representational system of visualization. Therefore, this ability eliminates the need for landmarks for navigation through space. In order to assess this particular ability, the test environment should minimize anything that can act as a landmark for the participants during the experiment. This type of spatial cognition has been of interest to our team for early diagnosis of Alzheimer's disease [76].

4.1. Experiment to Study Spatial Cognition

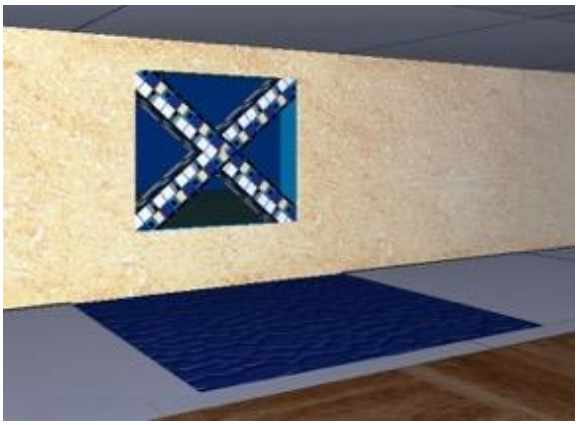
4.1.1. Implementation of the Virtual House

In this work the design of a VR experiment to assess spatial updating strategy and performance in a landmark-less environment is presented. This design is called the *Virtual House*. This design is based on a first person view simulation of an observer, who enters a three story cubic building. In order to target the egocentric ability of the spatial cognition, the design aimed to minimize landmarks (or perturbed orientation). In order the achieve this objective, the design of the house is symmetric from all the four sides. Each side of the house has an entrance and three windows on each floor. Each of the windows on the second and third floors can be selected as a target window.

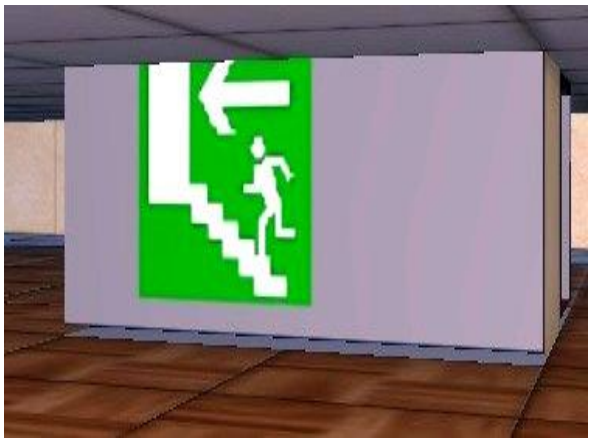
Out of the 24 windows (four sides, three windows on the second and third floors), one window is selected as the target window through a pseudo-random sequence. The selected target window is randomly marked with either, a triangle, a cross or a circle. Inside the house there are two staircases leading to the higher floors. The participant will be guided to the staircase by the sign shown on the sidewall of the staircase. Figure 4.1 shows three views of the *Virtual House*.



(a)



(b)



(c)

Fig. 4-1. Virtual House's, outdoor view (a), target window (b), staircase sign (c).

The *Virtual House* was implemented using Visual Studio C++ 2012 based on a setup (declared in *setup.h*) and loop (declared in *generic.h*) functions. The setup function loads the VR environment 3D models in random access memory (RAM) memory and initializes the participant's parameters

such as time, position and traversed distance. Next, the loop function is executed until an exit command is issued. The loop function runs various sub-functions for visual and auditory simulation as well as the physics engine. Figure 4.2 shows the hierarchy of the header used to implement various function of the *Virtual House*.

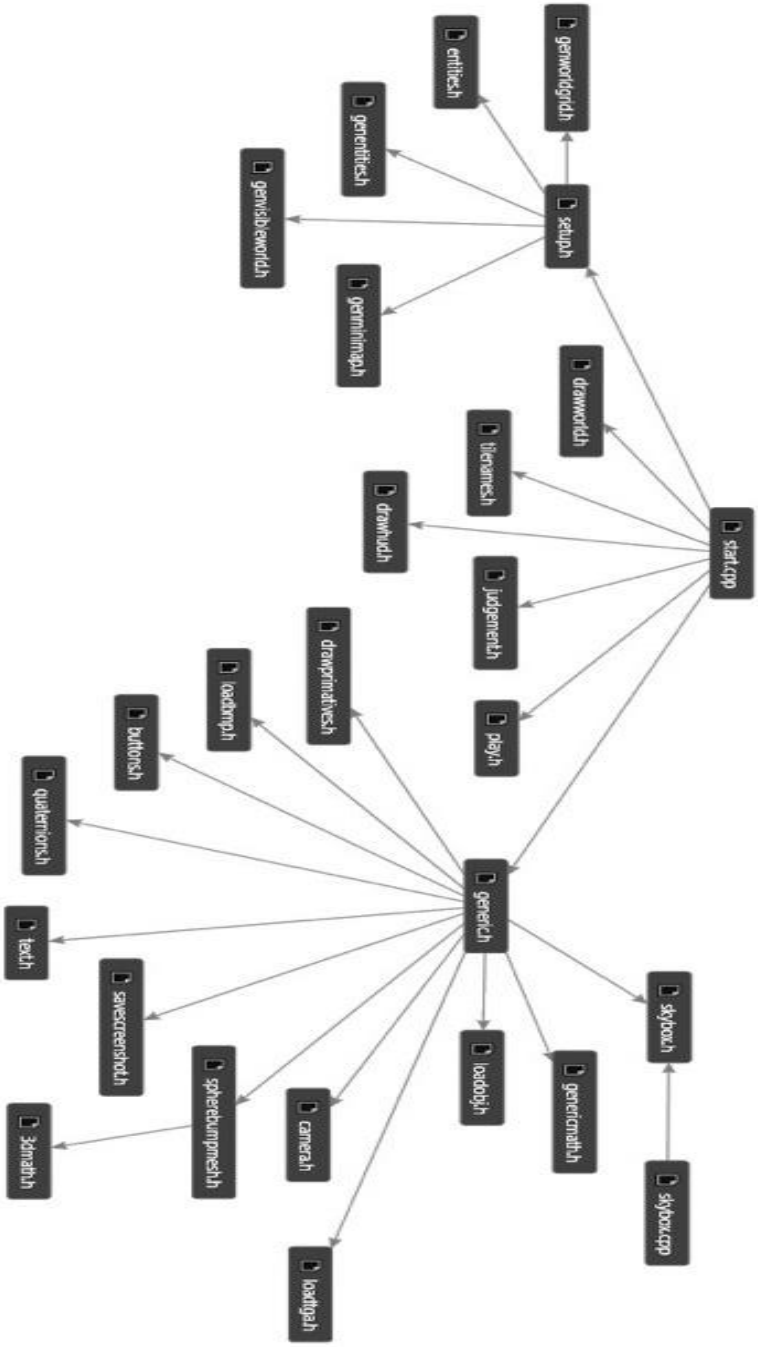


Fig. 4-2. Implementation hierarchy of the headers used in Virtual House.

While the VR engine is running, a log file collects the participant's parameters with a sampling rate of 1.0 kHz. The environment parameters include, elapsed time, navigation course (heading angle), position and traversed distance. In order to be able to modify the *Virtual House* with the least effort, the environment was designed with multiple modules. The *Virtual House* environment is comprised of fifty-seven 3D modeled objects with naturalistic textures. Forty-two objects are placed in fixed locations while fifteen objects are placed in variable locations. A variable location can be selected by a pseudo-random sequence at the beginning of each trial. That would make every trial unique in terms of the placement of the objects inside its VR world. The uniqueness of VR world in every trial reduces the learning effect for studies with multiple trials.

Figure 4.3 shows a typical trajectory of the displacement for one trial obtained from one of the study participants. The trajectory shows the locations visited by the participant from the starting point towards the target window.

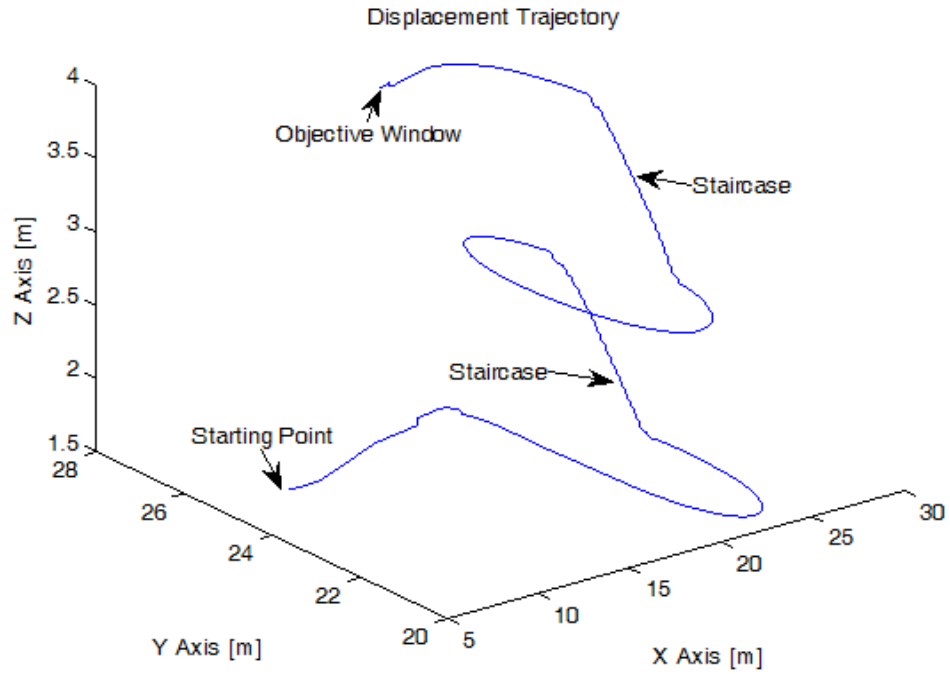


Fig. 4-3. Traversed trajectory in the virtual reality environment.

4.1.2. Design of the Virtual House Experiments

As it is described in 4.1.1, the *Virtual House* environment is comprised of a 3-story building with thirty-two windows which twenty-four of them (twelve windows for the 2nd and 3rd floor) can be assigned as a target window. Figure 4.4 shows the assigned letters for the twenty-four possible target windows. Initially, the experiments consived to cover all the possible twenty-four windows. However, due to practical issues and feedback from the participants, the trials were reduced to eight; thus, only eight pseudo-randomly chosen windows were selected as target windows. As the goal of the experiment was to challenge the spatial orientation ability, the middle target windows (total of six) were dropped from the sequence of trials (resulting in sixteen). Moreover, in an earlier study using the *Virtual House*, it was found that the side error (mixing up of the four building sides) was more significant that the corner errors (mixing up between left and right) [76]. Therefore, the trials were considered based on a pseudo-random sequence to cover every possible side window on the second and the third floor while the corner windows were randomly selected.

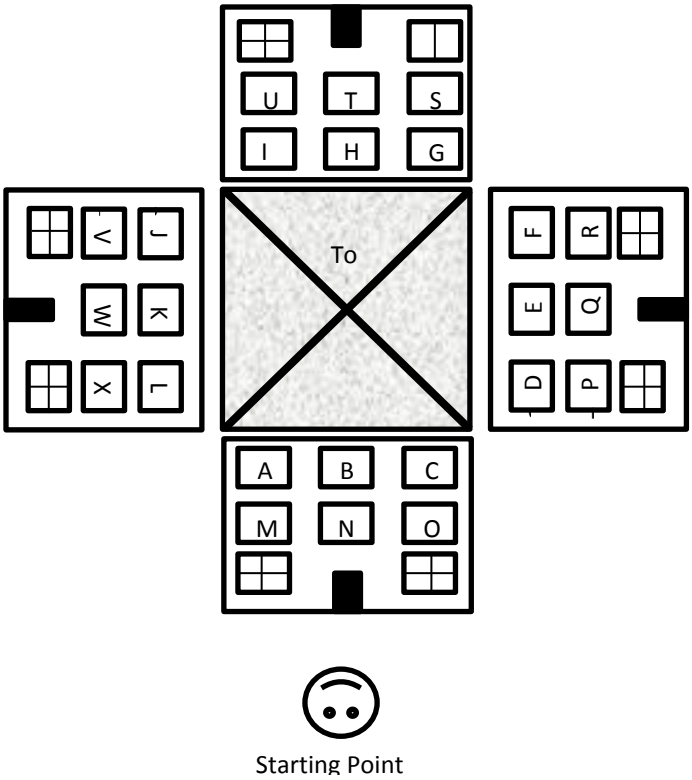


Fig. 4-4. Letter designation for possible target windows

4.1.3. Virtual House Practical Challenges

Initially the *Virtual House* experiment was performed using an ordinary consumer level joystick (Logitech Freedom 2.4 GHz) and a stationary display as it is shown in Fig. 4.4 Using this setup introduced two main challenges. First, about 30% of the participants, experienced motion sickness (similar to nausea) during or after participating in the experiment. Second, most of the older participants were not familiar with using the joystick to navigate through the VR environment. Hence, experience with using a joystick added a bias in their results. Thus, we stopped the experiment to design a method to resolve the motion sickness and computer-skill bias.

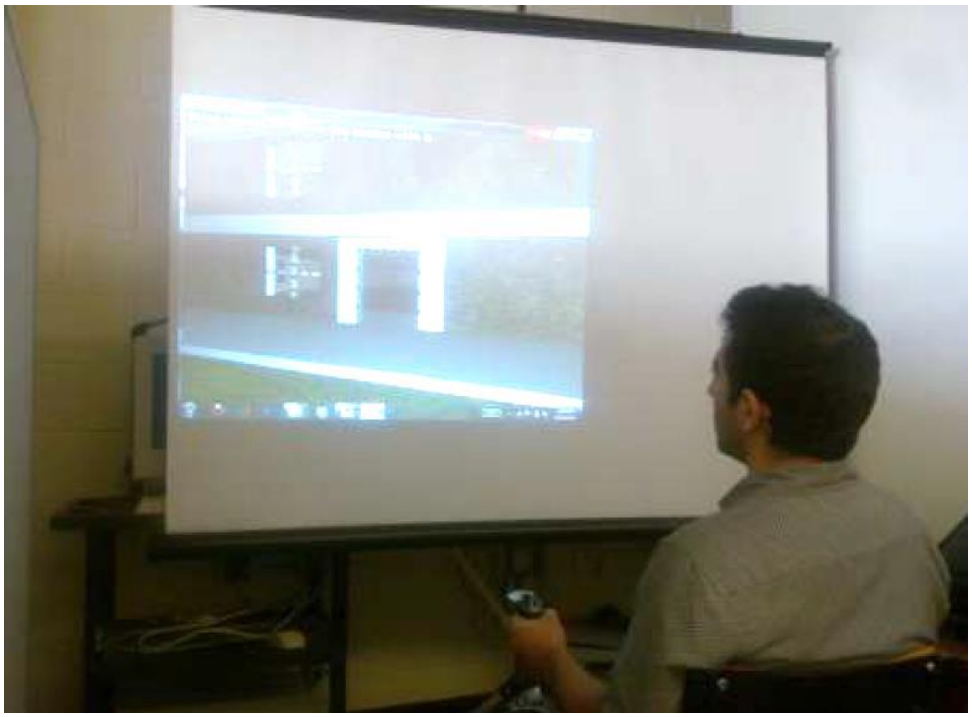


Fig. 4-5. Participant using a joystick and a stationary screen for interaction.

Motion sickness is caused by the disagreement between optical flow and physically experiencing motion; the disagreement could be the absence of one of the stimulations. In the case of using a computer screen, the participant perceives motion optically, while there is no physical experience of a motion to be perceived by vestibular system. This is the opposite of seasickness, where in the presence of physical motion experience there is no optical flow perceived by the person.

In addition to the motion sickness problem, using a Joystick as an input device can be challenging for people with the lack of computer experience; that can affect their performance in a test and bias the results. Therefore, our next aim was to design an intuitive input device to both minimize the motion sickness effect and remove the bias of computer knowledge or experience.

4.2. The VRNChair used as an Input Device

The technical design of the VR navigation chair (VRNChair) was explained in 3.5.1 The VRNChair was introduced as a solution for motion sickness and the bias caused from the lack of computer usage experience. The VRNChair is comprised from an ordinary wheelchair and a custom-made motion capture system. The motion capture measures the velocity of the two wheels of the wheelchair and estimates its overall motion. The overall motion is presented to the VR platform as an ordinary joystick input.

As shown in Fig. 4.5, the participant is asked to sit on the wheelchair or push it as a walker, and move the wheelchair physically in order to move in the VR environment. Using this method, the participant, physically experiences the motion that is perceived by the vestibular system almost in synchrony with the optical flow. Additionally, the participant's computer usage experience is not affecting their overall performance since they are moving physically to provide the navigation commands.

Fig. 4-6. VRNChair used with a laptop screen to interact with the virtual reality platform.

4.2.1. Using a Head Mounted Display (HMD) with the VRNChair

The VRNChair resolves the problem of motion sickness significantly but at the cost of the requirement of a physical space equal or greater than the size of the VR environment. However, moving in a physical environment increases the chance of distraction for the participant during the experiment and subsequently reduces the participant's immersion in the VR environment. This is because a laptop screen cannot cover the participant's peripheral vision.

In order to increase immersion in the VR environment and have the peripheral view coordinated with the optical flow, an HMD from the Oculus Rift Development Kit 2 (DK2) was employed. The HMD covers the peripheral vision of the participant during the experiment and increases the level

of immersion by reducing the chance of distraction from the surrounding environment. In addition, it allows the participant to freely move their head around and observe their surrounding in the VR environment. Figure 4.6 shows a participant wearing the HMD and using the VRNChair to navigate through the VR environment.

Fig. 4-7. Participant using the HMD and the VRNChair.

The HMD is equipped with a head tracking system based on an inertial measurement unit (IMU). The IMU uses a set of accelerometers, gyroscopes and magnetometers (AGM) to calculate the head angles (role, pitch and yaw), based on the earth gravity vector and the magnetic north pole. The result is an absolute measurement of heading with respect to a global system.

4.2.2. VRNChair and Measurement Drifting Issue

The VRNChair measures the translational ($\dot{X}_{VRNChair}$) and rotational ($\dot{\theta}_{VRNChair}$) velocities using the kinematic model of the wheelchair. The estimated translational and rotational velocities are reported to the host computer based on a constant time interval (δt). The heading angle of the VRNChair (θ_{chair}) in the VR environment is calculated from the integration of the reported rotational velocities and can be estimated as shown

$$\theta_{chair} = \int \dot{\theta}_{VRNChair} dt = \delta t \sum \dot{\theta}_{VRNChair} \quad (4.1)$$

Using integration to estimate the heading angle of the VRNChair results in relative measurement. Relative measurements are prone to drifting and additive error.

Since, the HMD uses absolute head tracking measurement and the VRNChair is only equipped with encoders, the heading is estimated using a relative measurement. As time passes, the relative measurement accumulates error and the measurement drift will cause a disagreement between VRNChair heading angle and the HMD head angle estimation. This will result in an unworkable combination of the HMD and the VRNChair.

In order to solve the drifting problem, the VRNChair 2.0 was developed. The VRNChair 2.0 is equipped with an IMU system in addition to the wheel encoders and the motion capture system. It fuses the measurement based on the encoders with the absolute measurement of the IMU system to assure accuracy and minimizes the drift effect of the heading angle estimation.

4.2.3. VRNChair and Sound Interference and Distraction

Since the VRNChair requires a physical place for the participant to navigate in, any external sound sources may act as undesired cues, and may affect the performance of the study; they can reduce the focus of the egocentric spatial experiment by adding unwanted landmarks. As such, they may have a negative impact and reduce the reliance of the participant on spatial orientation to solve a challenge.

This issue was resolved by adding audio rendering capabilities to the VR platform. The audio rendering simulates environmental noise based on the type of the environment the participant experiences in the VR. The environmental noise in an indoor environment changes based on the dimension and the material type, for the surroundings in the VR environment.

In order to have the environment noise more naturalistic, it was recorded with a looping capability. This eliminates disjoints caused from restarting the sound track when used to simulate the environmental noise. To improve the results, an acclaimed noise cancellation (ANC) headphone from Bose Quiet Comfort (QC-15) was selected to reduce the external noise effect and enhance the rendered sound.

4.3. Virtual Reality based Spatial Cognition Experiments

To evaluate the designed *Virtual House* and its different versions. Several few pilot studies were conducted that helped improve the experiment to its next version. The Biomedical Ethics Board of the University of Manitoba approved the study and its amendments, and all subjects signed an informed consent form prior to the experiments.

4.3.1. Studying the Effect of Aging on the Egocentric Spatial Cognition

The study of the effect of aging on the spatial cognition in a landmark-less environment was performed using the *Virtual House*. The test was performed on fifty-two cognitively healthy individuals (twenty-five females) with age range of 19-82 years. The experiment was performed at the VR center of the Industrial Technology Center (Winnipeg, Manitoba, Canada). The cognitive level of the subjects was tested by the Montreal Cognitive Assessment (MoCA) questionnaire [75], in which a score of equal or greater than 26 out of 30 is considered normal; all participants were within the normal cognitive range.

Before performing the test, the participants were allowed to practice moving around in the VR environment. After getting familiarized with moving in and out of the *Virtual House*, each participant performed the experiment (reaching an assigned target window) in sixteen different trials. The only instruction given to participants was to approach the target window using the shortest path that they perceived.

Before the participant started to navigate, the target window was shown to them by rotating the *Virtual House*, such that they were able to see the location of the target window from outside the house. After the first eight set of trials, the participants were allowed to rest for about ten minutes before starting the second set of trials (trials nine to sixteen). Each set of the eight trials covered the eight different possibilities of the target window location (two floors, each with four sides). It should be noted that the two sets of eight trials were identical in terms of assignment of the target window; for example, in each pair of trials, such as 1&9, 2&10, 3&11, etc., the target window was on the same side of the house and on the same floor.

The participant's performance error was measured by calculating the difference between the participant's traversed distance and the minimum distance required to reach the target window, which is calculated by the artificial intelligence based algorithm in the game engine. This difference is called the traversed distance error, and was normalized by the minimum desired distance in each trial. Finally, the error was normalized with respect to the maximum desired traversed distance and the median of the sixteen trials had been chosen as the indicator for the spatial cognition performance of the subject. The results of this pilot study was published in [77] and are presented in Section 5.1 of Chapter 5.

4.3.2. Studying the Effect of Using the VRNChair as an Input Device

The VRNChair was introduced to reduce the effect of motion sickness and computer usage experience bias. To evaluate the ability of VRNChair to resolve these two problems, a test was performed to compare the performances of cognitively healthy (MoCA >26) young and older adults, where the young group individuals have had computer gaming experience and the older adults did not. The young participants were 17 individuals (26.0 ± 5.2 years all < 40 year, all males), who were asked to perform the *Virtual House* test using a Joystick, and another group of 17 young individuals (27.6 ± 3.2 years, all < 40 y and all males) were asked to perform the same test using the VRNChair. All the participants were engineering graduate students at the University of Manitoba. This assured us that both young groups had the same exposure to computers and VR. The older participants were also in two groups: one group (65.1 ± 19.9 years, all males) were asked to perform the *Virtual House* experiment using a joystick, while the participants of the other group (68.5 ± 5.5 years, all males) were asked to perform the same test but using the VRNChair.

The reason for choosing two different groups was to reduce the chance of any plausible learning bias or the effect of order of performing the test with the joystick or VRNChair. The joystick used for the experiments was a Logitech ATK 3 (Logitech, 2012). For this pilot study, we used the same protocol of the experiment except that the number of trials were reduced from 16 to eight. The eight target windows were selected pseudo randomly as explained in Section 4.1.1.

Additionally, to study the effect of the VRNChair to reduce motion sickness a group of twelve of individuals in the age range of twenty to fifty-four (36.83±16.86 years, eight males) years were asked to perform the *Virtual House* test with both a joystick and the VRNChair. The resulting trajectories were spatially resampled (minimum distance of interest (MDI) of 10cm) and filtered (spatial low pass with spatial cut-off frequency of 50cm). Additionally, a questionnaire was used to ask participants' subjective feedback on plausible motion sickness. The joystick used for this experiment was a Logitech ATK 3 [78].

The traversed trajectory of study participants in the VR experiment was stored for further comparison between the use of joystick and the VRNChair. The game engine stores the absolute position of the participant in the VR environment with a 1.0 millisecond time interval. The absolute position is based on a fixed reference point inside the VR environment. It should be noted that using the VRNChair as an input device results in unwanted movements during the experiment in comparison to using the joystick. Thus, to have a fair comparison between the joystick and the VRNChair, both trajectories were spatially sampled and filtered.

4.3.3. Studying the Effect Using an HMD with the VRNChair

In order to study the effect of using an HMD with the VRNChair an experiment was performed [79]. The experiment for this study was based on four trials, toggling between laptop screen and the HMD. Each trial started by rotating the house in front of the participant with a particular target window marked with an 'X'. The participant was then instructed to navigate through the house to reach the target window. All participants were allowed to have one practice run prior to the start of the experiment to become familiar with the system. The study participants (18 individuals, 14 males, age = 25.8±8.7 years) were naive to the experiments.

After completing the assessment, participants were asked to rate their overall feedback of any motion sickness they might have felt on a scale of 0-4 (zero, for "feel fine" to four, for "extremely ill"), once immediately after the experiment and then again one hour after completing the test in order to check any residual or delayed-onset effects of motion sickness. The order of the trials started by using the laptop screen and toggled to the HMD for the next trial. This order was selected to control and justify for any plausible learning effects. Since, the focus of this experiment was to find the differences in the feeling of motion sickness as well as between the obtained trajectories from the trials with and without an HMD, the target window was set to the same location for each of the four trials.

4.4. Summary

In this chapter, the experiments used to validate the implementation on the VR platform were introduced. The first experiment was based on an ordinary joystick as the input device to receive the navigation commands from the participant. The second experiment used the VRNChair and the third experiment investigated the effect of using a HMD on the overall results of the experiment using the VR platform.

Chapter 5

Results and Discussion

Three sets of experiments were conducted to evaluate the virtual reality (VR) house under various versions: investigating the effect of aging on spatial cognition performance, comparing the VRNChair with an ordinary joystick as an input device, and one for comparing a head mounted display (HMD) version with the laptop screen display. The results of the three experiments are provided in the following sections in the order described above.

5.1.Result of the Effect of Aging on the Egocentric Spatial Cognition

The results of the experiment described in 4.3.1 is shown in Fig. 5.1 by plotting the normalized performance error of the participants versus their age. The result shows a linear positive relationship between spatial error (difference between participant's traversed distance and the minimum require distance) and aging, which is congruent with the literature showing a decline of spatial ability with aging.

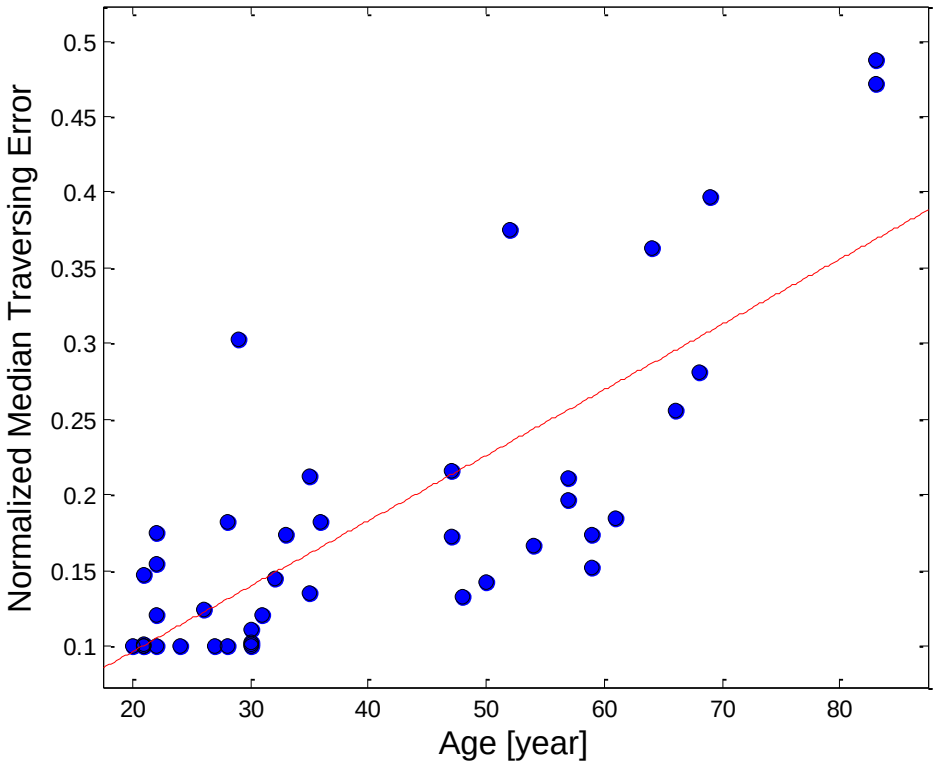


Fig. 5-1. Median value of the normalized performance error of the subjects versus their age.

Figure 5.2 shows the normalized traversed distance error averaged between the participants in three different age groups (< 40 years, 40 < age < 60 and 60+ years) for each trial. This plot demonstrates if there is any learning effect due to repetition of the experiment. Note that the starting point and the target window of trials 1&9, 2&10, 3&11, etc., are the same. Based on the results, the learning effect is not substantial in the (< 40 years) age group while it is more visible in older participants.

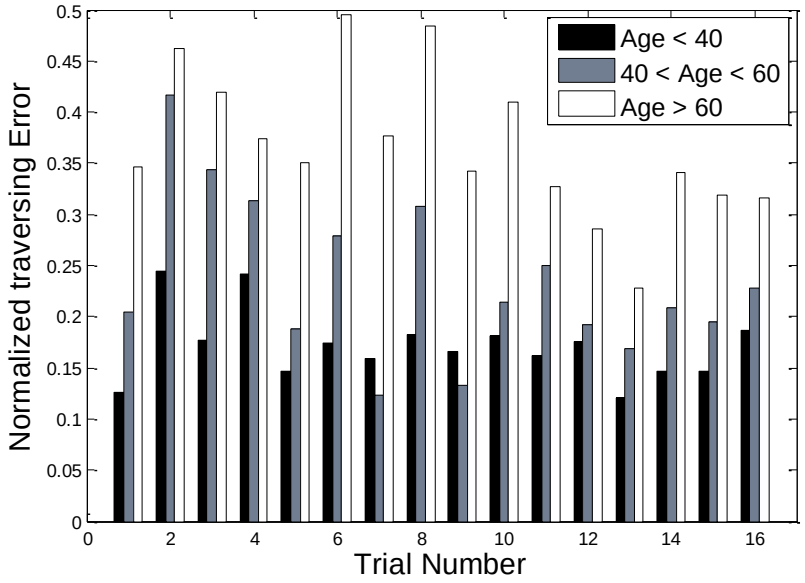


Fig. 5-2. Normalized error of the 16 trials averaged between the subjects for different age groups.

Discussion on the Results based on the Joystick

The results of this pilot study shows that a VR based experiment can be used to evaluate human's spatial cognition, in particular the egocentric orientation using the *Virtual House*. Literature hypothesizes that the egocentric spatial cognition declines with aging [4]. The results from the *Virtual House* experiment show a direct relation been aging and the participants decline in performance in the test.

The *Virtual House* is designed based on symmetric cubic house to minimize the effect of landmarks. However, it can be argued that certain unique objects in the environment can act as landmarks. Thus, using a symmetric approach in the layout reduces the unwanted use of the unique objects as landmarks. For instance, the staircase can be duplicated while placed in the middle of each floor. Additionally, the doors on the first floor on each side were to avoid the existence of the door to act as a landmark. Additionally, the number of windows in each side of the house and on

the two floors were the same.

In the beginning of each trial, the target window was shown to the participants from outside of the house by rotating the house 360° degrees (i.e., the house with the target window will be rotated in front of the observer). At this point, the participant was expected to remember where the target window is located, enter the house and reach the target window inside the house.

If a participant becomes confused and unable to find the target window in the *Virtual House* and begins to find it by trial and error, the traverse distance will increase. This resulted in a higher difference between the traversed distance, the minimum required distance and subsequently a lower test performance. The result in Fig. 5.1 clearly shows that the performance error increased by age almost linearly. This is congruent with previous studies [80], where 86% percent of younger participants were able to reach the target, while only 24% of the older subjects were able to succeed [1].

In order to investigate whether there was a learning effect because of repetitive trials, the performance error of the participants was calculated for every trial. As can be seen in Fig. 4, there is no substantial trend of learning throughout the trials for the (< 40 years) participants. However, older participants (60+ years) show learning effect by improvement over repetition of trials. It is speculated that this effect is related to the computer experience level of the participants.

5.2. Results of Experiment Using the VRNChair as an Input Device

Among the two groups of young participants (< 40 years) that performed the *Virtual House* test using a joystick and the VRNChair, no one reported any signs of motion sickness. They showed the ability to navigate through the VR environment. The results indicate that the participants

performed similarly when using the VRNChair or a joystick as the input device. The average of the traversed distance, the time to reach the target window and the spatial error (calculated from the different between the minimum distance and the traversed distance) using the joystick and VRNChair are shown in Table 5.1 This experiment showed that young participants performed very similar using either the VRNChair or joystick in the VR test.

Table 5-1: Results of the experiment on young participants using the Joystick vs. using the VRNChair

	Joystick	**VRNChair**
Distance (metres)	41.8 ± 8.95	39.28 ± 6.15
Time (seconds)	38.8 ± 10.3	47.3 ± 12.4
Spatial Error (meters)	8.5 ± 6.4	7.5 ± 7.1
Average Velocity (m/s)	1.07	0.83

A similar study was performed on older individuals (60+ years) with the *Virtual House* test using a joystick and the VRNChair. Unlike the results of the young participants, the older participants showed a significant improvement in terms of traversed distance error (the difference between the actual and the calculated minimum distance) when using the VRNChair as the input device (from 51.74 ± 8.33 to 33.24 ± 5.25 meters). Table 5.2 shows the average of the traversed distance, the time to reach the target window and the spatial error.

Table 5-2: Results of the older participants using the Joystick versus using the VRNChair

	Joystick	**VRNChair**
Distance (metres)	76.35 ± 8.34	59.04 ± 5.17
Time (seconds)	142.89 ± 10.27	143.28 ± 10.06
Traversed distance error Error (meters)	51.74 ± 8.33	33.24 ± 5.25
Average Velocity (m/s)	0.53	0.41

For both young and older participants, the time required to finish the task using the VRNChair was higher. Considering the traversed distance, it can be seen that the average velocity using the VRNChair is less than the joystick in both age groups of the participants. This may attribute to the

fact that unlike the joystick, navigation using the VRNChair requires physical movement. It can be clearly seen that older participants navigated slower than the younger ones on average. More importantly, as can be seen in Table 5.2, the older participants made significantly lesser errors when used VRNChair while the error of the young participants using joystick and VRNChair was similar. The result agrees with the hypothesis that the lack of computer experience adds an unwanted bias to the results.

Four out of the ten older participants using the joystick experienced motion sickness during or after the *Virtual House* test. By contrast, only two out of the 10 participants using the VRNChair felt a minor level of motion sickness. However, those two using the VRNChair experienced distraction during the test. We hypothesized that the distraction was due to asynchrony between the participants' peripheral vision, and what was seen on the screen during the VRN experiment. Hence, the subsequent experiment considered to use the HMD with the VRNChair. However, using the HMD with the VRNChair required technical modifications in the experimentation system.

5.3. Results of Experiment Using an HMD with the VRNChair

In order to investigate the effect of using a HMD with the VRNChair, two different analyses were performed on the data collected from the experiment [79]. The test participants for this investigation were primarily young students and academics from the University of Manitoba (18 individuals, 25.8 ± 8.7 years, 14 males). This group consisted predominantly of males between the ages of 20 to 25 years (11 individuals). Furthermore, 12 out of 18 participants reported that they had experience playing first person shooter (FPS) games. Prior experience with immersive games like this, even without an HMD, could cause someone to adapt to the vestibular-visual sensory

conflict, causing a predisposition to reduced levels of motion sickness [81]. Since this experiment is not interested in analyzing the spatial orientation of participants, the participants were guided to find the correct target window. The same target location was re-used for each of the four trials, a training trial was provided, and advice was given if asked for. However, in a limited number of trials, some participants still entered an incorrect room; these erroneous trials were discarded from analysis. The excluded trials are summarized in Table 5.3.

Table 5-3. The Summary of Discarded Trials

Platform	Erroneous Trial 1	Erroneous Trial 2	Total
LCD	10	3	13
HMD	5	2	7

Note that Table 5.3 does not include data on the training trial, which occurred immediately before the first Laptop screen trial. Note also that participants were more likely to have an error during the first LCD trial than the first HMD trial; however, this could be due to a learning effect. With a larger sample, this could have been controlled by splitting subjects into two groups, where one group would start with the laptop screen and the second group would start with the HMD.

In addition to these cases, there were seven cases where participants took an unnecessary detour (such as looping around the stairs) to get to the target window, even though they did not enter an incorrect room. These trajectories were omitted because they represent a different route and would tend to skew the final results. Conclusively, seven cases could not be used due to testing error. This left 21 trials where the HMD was used and 16 trials where the laptop screen was used. The 37 qualifying trial trajectories are superimposed and shown in Fig. 5.3.

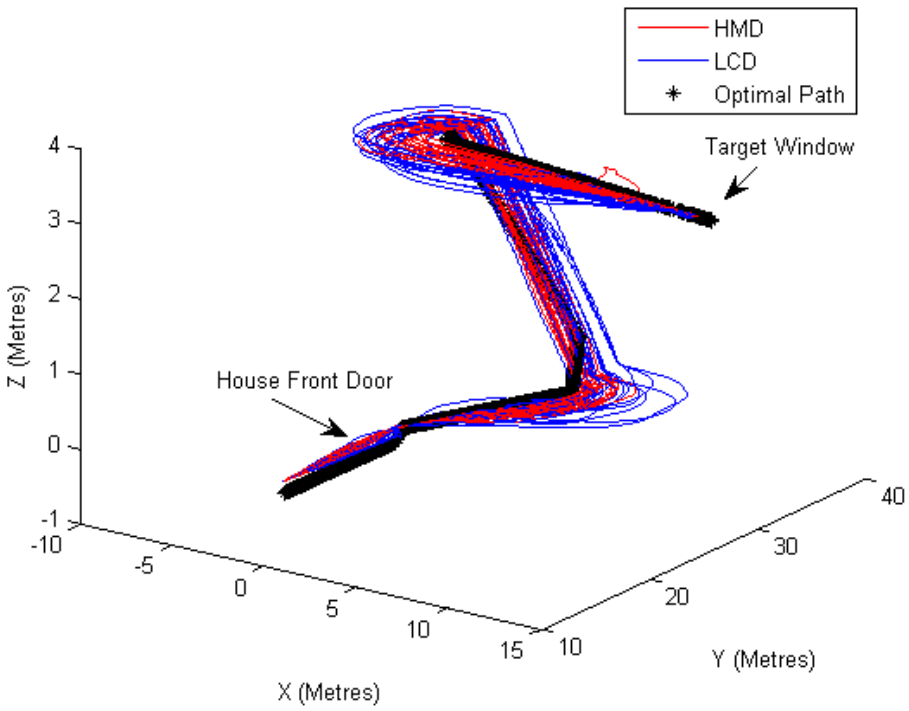


Fig. 5-3. Trajectories obtained from the trials using the laptop screen versus the HMD [79].

It can be seen that the trajectories taken by people wearing the HMD were the closest match to the optimal trajectory, and had less variation than the laptop screen trajectories. The distances and solution times are summarized in Table 5.3.

Table 5-4. Results of traversed distance and time for trials using the HMD vs. using the laptop screen

	HMD	**Laptop Screen**
Distance (metres)	56.0 ± 3.8	61.3 ± 5.5
Time (seconds)	61.8 ± 14.3	59.0 ± 18.0

Since 16 trajectories from the experiment with the laptop screen were found to be eligible for comparison, the 16 corresponding HMD trajectories were used to perform a matched-pairs t-test. It was found that the paths taken by people using the HMD were significantly ($p < 0.01$) shorter than the paths taken by people using the laptop screen. There was no significant difference between the times taken to reach the target while wearing the HMD and the times taken to reach the target

while looking at the laptop display ($p = 0.53$). This indicates that people move most slowly and carefully while viewing the VR environment through an HMD oppose to viewing the VR environment through the laptop screen.

Furthermore, learning effects were investigated by means of a second matched-pairs t-test comparing the first round of laptop screen trials with the second round of laptop screen trials. The first round of HMD trials was also compared with the second round of HMD trials. No significant difference was found between distances or the times, although only four participants had all of their trials qualify. This suggests that learning effects were not significant.

Assessing levels of motion sickness was a secondary objective of this work, and was done by having participants subjectively rank their sickness on a scale of zero (feel fine) to four (extremely ill). The average score for motion sickness was 0.42/4.

In this study, five of the 18 participants reported experiencing very mild headache/nausea after the test, and two of these reported very mild symptoms approximately one hour after the test. One out of 18 participants reported relatively severe headache/nausea immediately after the test and one hour after completing the test (this person reported being particularly susceptible to motion-sickness, even while riding in cars and buses).

One person (that was not part of this study) that had experienced severe motion sickness with our previous setup described in [72] [80] performed a trial run using only the HMD (no laptop screen) and reported no motion sickness. This suggests that some of the motion sickness reported by users may have been caused by the laptop screen trials and not by the HMD trials. Due to the relatively low occurrences of motion sickness in participants and the small sample size, further work will need to be done to find whether using the HMD reduces motion sickness as compared to using the laptop screen.

In order to reduce motion sickness in people using their HMDs, Oculus VR provides a calibration tool to set the HMD to match a particular person's interpapillary distance (IPD). The effectiveness of this measurement was illustrated to us first hand: in one trial, we neglected to select a user's calibration profile, and the user became ill almost instantly. After we corrected the error, the participant took a short (~ 2-minute) break and wished to continue with the assessment. The participant reported that the IPD-calibrated VR environment was less sickness inducing, and they were able to complete the assessment.

Motion sickness effects have been reported in [82], [83], [84], and [85]. However, only [84] presented the number of participants that were severely affected: they had 10% of participants withdraw from their study; in our study, no one withdrew from our experiment due to motion sickness. The large number in [84] may be due in part to the fact that participants were older adults, and consisted of an even distribution of males and females. They postulated that women are more sensitive to motion sickness than men are, so since our study involved mostly males, a lower level of motion sickness could be expected. [83], [84], and [85] presented their motion sickness data as an aggregate score and an average score only; all groups used different scoring techniques that considered different factors, making it impossible to compare them quantitatively.

Almost every participant reported discomfort or experienced a "weird feeling" when ascending and descending the central ramp. This is because in real life, one would expect resistance due to gravity, not to mention an acceleration component in the vertical direction. Since these cannot easily be re-created in our VR platform, their absence may contribute to user discomfort. A possible solution is to use a slow-moving elevator in place of the ramps.

In summary, adding the HMD to the VRNChair for the *Virtual House* experiment reduced motion sickness and provided more immersion resulting in more accuracy.

5.4. Summary

In this chapter, the results obtained from the experiments were presented. The first experiment showed the relationship between the participant's performance versus their age. However, it was noticed that the computer usage experience as well as motion sickness could affect the overall performance of the participant. The second experiment showed the effect of using the VRNChair versus using an ordinary joystick. The results showed that elderly participants perform better with VRNChair. The third experiment investigated the usage of an HMD, and the results showed that user was able to use the HMD as an alternative to interact with the VR platform.

Chapter 6

Conclusion and Recommendations for Future Work

This thesis investigated the design of a Virtual Reality (VR) environment for neuroscience type studies. While the VR development procedure is general and applies for any application, the fact that the motivation of this study was to design a VR experiment for investigating spatial cognition in older adults, posed several challenges that were addressed one by one in this thesis. The main challenges were motion sickness caused by the input device, bias of computer knowledge and the need for increased immersion to provide a naturalistic environment that is of particular importance when studying older adults with some degrees of cognitive decline. Overall, the evaluation results of this study show that the designed VR house can be used in spatial cognition studies as an alternative for physical test environments. Increased immersion can be achieved by using modern rendering techniques, higher definition 3D models, textures and a naturalistic game engine. The naturalistic game engine should include several physics effects such as gravity, friction, inertia, etc. Lighting can play a key role in the overall appearance of the objects as well as defining their properties. This thesis has provided a basis for development of such a design.

In addition to the visual rendering, sound rendering is also important. Using the binaural hearing, humans are able to localize the position of a sound source. Moreover, the overall appearance, geometry and shape of an environment causes reflections and changes in the environment sounds. These effects can give the perceiver an understanding about the surrounding environment. A sound rendering system can mimic this effect and increase the level of immersion for the participant in a VR environment.

Preliminary studies on the spatial cognition using the VR environment showed a relationship between aging and the decline of egocentric spatial cognition abilities. However, it was found that due to the wide age group (19-82), not all participants shared similar levels of computers usage experience. Hence, the results of those with lesser computer exposure was biased negatively do to their unfamiliarity with computers, specifically using a joystick. Additionally, a considerable number of the participants (30%) were experiencing motion sickness during and after performing the VR test.

As a remedy for both issues, the VRNChair was developed. The VRNChair is based on an ordinary wheelchair, equipped with a custom-made motion capture system. The motion capture system is capable to report the overall movement of the wheelchair to the computer. This setup is used as an alternative to the joystick system to reduce the effect of motion sickness and the bias of computer usage experience.

Initially the participants were asked to sit or use the VRNChair as a walker, looking into a laptop screen resting on the VRNChair (as it was shown in Fig. 4.4.). The number of participants, experiencing motion sickness was reduced and the computer usage experience bias was reduced or eliminated. However, the participants were more prone to distraction as the VR environment did not cover their peripheral vision. Additionally, physical sound sources were able to act as unwanted landmarks during spatial orientation experiments, where the participant is expected to rely solely on the egocentric spatial skills.

By adding the head mounted display (HMD), the participants' peripheral vision was covered. This resulted in less distraction while navigating the VRNChair through the physical environment.

Furthermore, using an acclaimed noise canceling (ANC) headphone and sound rendering reduced the effect of unwanted sound landmarks in the VR environment.

Due to the fact that the HMD uses an inertial measurement unit (IMU) sensor (absolute measurement), while the VRNChair uses wheel incremental encoders (relative measurement) to detect the heading angles, drift discrepancy between the two measurement can occur. This can result in a mismatch between the heading angle of a participant in the VR environment and their movement direction. The outcome will lead into an unusable system and hinders the experiment from progressing further.

To overcome this problem an additional inertial measurement unit (IMU) system was added to the VRNChair. The new system fuses the data acquired from the encoders and the IMU. The fusion of the two sensor eliminates the measurement drift in the heading estimation caused by the encoders on the VRNChair while it provides translation information using the encoders.

The VR software and input device constructed in [21] were extended to make use of an HMD. We compared people's path finding in a VR environment viewed through a laptop screen and through an HMD. It was found that people's navigational trajectories were more accurate when using the HMD than using the laptop screen, but solution times remained constant. This suggests that increased immersion in a VR environment results in an improvement in path finding, and more careful navigation. We found that a subset of users experienced minor motion sickness despite the vestibular stimulation offered by our custom input device, the VRNChair. These results indicate that we can begin integrating the HMD into longitudinal studies.

Finally, the followings can be considered:

- VR has the potential to be used for spatial cognition studies
- Intuitive interaction methods can improve fidelity of VR based experiment
- VRNChair can solve or reduce both the unwanted computer experience bias and motion sickness
- A head mounted display can improve the immersion in the VR environment

6.1. Recommendations for Future Work

The VRNChair is capable to resolve the issues caused by the lack of computer usage experience and the motion sickness with the cost of requiring a system a physical space for navigation. The motion sickness is resolved by allowing the users to physically move and experience the motion as seen in the VR. Therefore, the physical environment required for the VRNChair needs to be similar in size with the VR environment. This makes experiments with a large VR environments challenging, as they require a large physical space.

Recent advancement is micro electro mechanical systems (MEMS) made it possible to produce low cost inertial measurement units (IMU) with high accuracy. This technology made it possible to build a full body motion tracking system using an array of IMU sensors (17 units). Such a system will not require any reference and can be suitable for usage on the VR system presented in this dissertation.

Preliminary tests on using the VRNChair as a walker (instead of sitting on the VRNChair), revealed issues caused by gravity impact during gait steps. The gravity impact is received by the

HMD as well as the participant's vestibular system. Both, the participant's eyes and the HMD try to compensate for gravity impact, which results in overcompensation, causing the lack of gaze stabilization. A temporary solution used in this work was to ask the participants to sit in the VRNChair. A further development may address the following issue by investigating the threshold of the gravity impact and the eye stabilization mechanism.

Recent studies show the possibility of using galvanic vestibular stimulation (GVS) to eliminate the effect of motion sickness caused from using VR. In fact, a newer system can benefit from the GVS system with an intuitive method to navigate through the VR environment without the need for a physical space for the VRNChair.

6.2. Contributions of the thesis

This thesis investigated the three main research questions, and addressed them by designing a novel VR navigational environment and running experiments for its evaluation. The three main contributions of this thesis as described as following:

- The development of a game engine (VR engine) tailored for spatial cognition studies
    - Allowed experimentation with different hardware and real-time algorithms to benchmark patient's performance
- The development of a new interaction method with the VR environment (VRNChair) (including 3 generations of the VRNChair)
    - Reduce the unwanted bias caused from level of computer experience
    - Reduce motion sickness caused when using the VR environment
- The development of trajectory spatial resampling which made it possible to:
    - Spatial filter a temporal equidistance trajectory
    - Spatially compare two given trajectories

References

[1] J. Psotka, "Immersive training systems: Virtual reality and education and training," *Instructional Science,* vol. 23, no. 5-6, pp. 405-431, 1995.

[2] R. S. Haluck and T. M. Krummel, "Computers and Virtual Reality for Surgical Education in the 21st Century," *Archives of Surgery,* vol. 135, no. 7, pp. 786-792, 2000.

[3] A. A. Rizzo, M. Schultheis, A. K. Kerns and C. Mateer, "Analysis of assets for virtual reality applications in neuropsychology," *Neuropsychological Rehabilitation: An International Journal,* vol. 14, no. 1, pp. 207-239, 2008.

[4] Y.-H. Qiu, K. Hu and X.-J. Luo, "Application of Computer Virtual Reality Technology in Modern Sports," in *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*, Hong Kong, 2013.

[5] A. Vilar Soares Calado, M. Márcio Soares, F. Campos and W. Correia, "Virtual Reality Applied to the Study of the Interaction between the User and the Built Space: A Literature Review," in *Design, User Experience, and Usability. User Experience in Novel Technological Environments*, Berlin Germany, Springer, 2013, pp. 345-351.

[6] J. M. Zheng, K. W. Chan and I. Gibson, "Virtual reality," *Potentials, IEEE,* vol. 17, no. 2, pp. 20-23, 1998.

[7] H. Sauzéon, A. P. Prashant, L. Florian, W. Gregory, D. Marie, Z. Xia, G. Pascal and N. Bernard, "The use of virtual reality for episodic memory assessment: effects of active navigation," *Exp Psychol,* vol. 59, no. 2, pp. 99-108, 2011.

[8] M. J. Kim, W. Xiangyu, L. P. E. D., L. Heng and K. Shih-Chung, "Virtual reality for the built environment: a critical review of recent advances," *Journal of Information Technology in Construction 18,* vol. 2, pp. 279-305, 2013.

[9] R. Desai Parth, D. Pooja Nikhil, A. Komal Deepak and M. Khushbu, "A review paper on oculus rift-A virtual reality headset," *arXiv,* vol. 1408.1173, 2014.

[10] F. S. Fitzgerald, American Literature, 2008, p. 3.

[11] M. Zyda, "From visual simulation to virtual reality to games," *Computer,* vol. 38, no. 9, pp. 25-32, 2005.

[12] J. Taylor, "The Emerging Geographies of Virtual Worlds," *Geographical Review,* vol. 87, no. 2, pp. 172-192, 2010.

[13] A. S. Garcia, A. Olivas, J. P. Molina, J. Martinez, P. Gonzalez and D. Martinez, "An Evaluation of Targeting Accuracy in Immersive First-Person Shooters Comparing Different Tracking Apporaches and Mapping Models," *Journal of Universal Computer Science,* vol. 19, no. 8, pp. 1086-1104, 2013.

[14] R. Pausch, D. Proffitt and G. Williams, "Quantifying immersion in virtual reality," in *SIGGRAPH '97 Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, Los Angeles, 1997.

[15] B. Landau and R. Jackendoff, ""What" and "where: in spatial language and spatial cognition," *Behavioral and Brain Sciences,* vol. 16, no. 2, pp. 217-265, 1993.

[16] E. Coluccia, A. Bosco and M. A. Brandimonte, "The role of visuo-spatial working memory in map learning: new findings from a map drawing paradigm," *Psychological Research,* vol. 71, pp. 359-372, 2007.

[17] S. Garden, C. Cornoldi and R. H. Logie, "Visuo-Spatial Working Memory in Navigation," *Applied Cognitive Psychology,* vol. 16, pp. 35-50, 2002.

[18] W. Wen, T. Ishikawa and T. Sato, "Individual Diferences in the Encoding Process of Egocentric and Allocentric Survey Knowledge," *Cognitive Science,* vol. 37, pp. 176-192, 2013.

[19] C.-N. Carolina, "Surround-screen projection-based virtual reality: the design and implementation of the CAVE," in *SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques* , Anaheim, CA, USA, 1993.

[20] M. Lewis and J. Jacobson, "Game Engines," *Communications of the ACM,* vol. 45, no. 1, pp. 27-31, 2002.

[21] A. Byagowi, D. Mohaddes and Z. Moussavi, "Design and Application of a Novel Virtual Reality Navigational Technology (VRNChair)," *Journal of Experimental Neuroscience,* vol. 8, pp. 7-14, 2014.

[22] R. P. Darken, W. R. Cockayne and D. Carmein, "The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds," in *Proceedings of UIST*, Banff, Canada, 1997.

[23] H. Iwata, "Walking about virtual environments on an infinite floor," in *Virtual Reality, 1999. Proceedings., IEEE*, Houston, TX, 1999.

[24] P. Milgram, H. Takemura, A. Utsumi and F. Kishino, "Augmented Reality: A class of displays on the reality-virtuality continuum," in *Proceedings of Telemanipulator and Telepresence Technologies*, 2007.

[25] D. Cohen, "Cathode-Ray Tube Amusement Device – The First Electronic Game," *AboutTech,* 2012.

[26] S. L. Kent, The first Quarter: A 25-Year History of Video Games, BWD Press, 2000, p. 83.

[27] H. Newman, "Games poised to outstrip broadcast TV revenues, SuperData finds," *VenureBoat,* 2015.

[28] W. Mark J.P., "The Video Game Explosion: A History from PONG to Playstation and Beyond," *ABC-CLIO,* p. 259, 1984.

[29] C. Crawford, "A Taxonomy of Counter Games," *The Art of Computer Game Desing,* pp. 8-11, 1982.

[30] R. H. Bear, "How Video Games Invaded the Home TV Set," 2014.

[31] S. Berberich, "Video games starting to get serious," *Gazette,* 31 Augut 2007.

[32] J. Strickland, "How Virtual Reality Military Applications Work," *Sience, HowStuffWorks,* 24 11 2009.

[33] R. Hackathorn, "Serious Games in Virtual Worlds: The Future of Enterprise Business Intelligence," *Search Business Analytics,* 21 03 2007.

[34] J. Ward, "What is a Game Engine?," *GameCareerGuide,* 29 04 2008.

[35] J. Gregory, Game Engine Architecture, Second Edition, New York: CRC Press, 2015, pp. 12-13.

[36] A. Michael, "Ramblings in Realtime," 22 08 2012.

[37] D. Trenholme and P. S. Shamus, "Computer game engines for developing first-person virtual environments," *Virtual Reality,* vol. 12.3, pp. 181-187, 2008.

[38] A. Maciel, H. Tansel, L. Zhonghua, P. N. Luciana and D. Suvranu, "Using the PhysX engine for physics-based virtual surgery with force feedback," *The International Journal of Medical Robotics and Computer Assisted Surgery 5,* vol. 3, pp. 341-353, 2009.

[39] C. Li-Keng, C. Ming-Hua and C. Wei-Hua, "Measuring virtual experience in a three-dimensional virtual reality interactive simulator environment: a structural equation modeling approach," in *Virtual Reality (18)*, 2014.

[40] P. Milgram and F. Kishino, "A taxonomy of Mixed Reality Visual Displays," *IEICE Transactions on Information Systems,* Vols. E77-D, no. 12, pp. 1321-1329, 1994.

[41] M. Billinghurst, H. Kato and I. Poupyrev, "The MagicBook - moving seamlessly between reality and virtuality," *Computer Graphics and Applications, IEEE,* vol. 21, no. 4, pp. 6-8, 2001.

[42] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre, "Recent advances in augmented reality," *Computer Graphics and Applications, IEEE,* vol. 21, no. 6, pp. 34 - 47, 2001.

[43] Company, Ford Motor, "Ford- Cars, SUVs, Trucks & Crossovers | Ford Vehicles | The Official Site of Ford Vehicles | Ford.com," Ford Motor Company, 10 October 2013. [Online]. Available: http://www.ford.com/crossovers/edge/features/Feature15/. [Accessed 10 October 2013].

[44] Z. Szalavari, D. Schmalstieg, A. Fuhrmann and M. Gervautz, "“Studierstube”: An environment for collaboration in augmented reality," *Virtual Reality,* vol. 3, no. 1, pp. 37-48, 1998.

[45] H. Regenbrecht, T. Lum, P. P. Kohler, C. Ott, M. Wagner, W. Wilke and E. Mueller, "Using Augmented Virtuality for Remote Collaboration," *Presence,* vol. 13, no. 3, pp. 338-354, 2004.

[46] D. C. Cui, "The Smart Glasses Revolution & The Rise of Augmented Reality in Manufacturing," *Vuzix view the future,* 2014.

[47] P. Travers, "The History and Incredible Potential of Wearable Dislplays," *Vuzix view the future,* pp. 7-8, 03 2014.

[48] K. Pal Rohit and S. Nitish Kumar, "Optical head mounted display using augmented reality," *International Journal of Research in Engineering and Applied Sciences,* vol. 4.7, pp. 1-10, 2014.

[49] R. Desai Parth, D. Pooja Nikhil, A. Komal Deepak and M. Khushbu, "A review paper on oculus rift-A virtual reality headset." arXiv preprint," *arXiv preprint arXiv,* vol. 1408.1173, 06 08 2014.

[50] Iribe, "Over 100,000 Oculus Rift DK2s Shipped," 16 02 2015. [Online]. Available: http://www.vrfocus.com/2015/02/iribe-100000-oculus-rift-dk2s-shipped/.

[51] L. Palmer, "Palmer Luckey Explains Oculus Rift's Constellation Tracking and Fabric," *VRFocus,* 17 06 2015.

[52] D. H. Eberly, "3D game engine design: a practical approach to real-time computer graphics," *CRC Press,* 2006.

[53] G. Jason, Game Engine Architecture, CRC Press, 2009.

[54] T. Alan, Game Engine Design and Implementation, Jones & Bartlett Publishers, 2011.

[55] D. Regan and I. B. Kenneth, "Binocular and Monocular Stimuli for Motion in Depth: Changing-disparity and Changing-size feed the same motion-in-depth stage," *Vision Reseach,* vol. 19.12, pp. 1331-1342, 1979.

[56] S. B. Niku, Introduction to Robotics: Analysis, systems, applications., vol. 7, Prentice Hall, 2001.

[57] A. Gupta, S. Mazumdar and S. Choudhry, Practical Approach to Ophthalmoscopic Retinal Diagnosis, vol. 4, Jaypee Brothers Publishers, 2014.

[58] PhysicsClassroom, [Online]. Available: http://www.physicsclassroom.com/class/refrn/Lesson-5/The-Mathematics-of-Lenses.

[59] N. Whiting, "Integrating the Oculus Rift into Unreal Engine 4," *The Art & Business of Making Games,* 06 11 2013.

[60] G. Henri, Continuous shading of curved surfaces. Seminal Graphics: Pioneering efforts that shaped the field, In Rosalee Wolfe: ACM Press, 1998.

[61] D. Begault R., "3-D Sound for Virtual Reality and Multimedia," NASA Ames Research Center, Moffett Field, CA, 2000.

[62] L. Jeffress, "A place theory of sound localization," *Journal of Comparative and Physiological Psychology,* no. 41, pp. 35-39.

[63] C. P. Brown and O. D. Richard, "An efficient HRTF model for 3-D sound," *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. New York: IEEE,* pp. 298-301, 1997.

[64] T. Chong-Jin and G. Woon-Seng, "User-defined spectral manipulation of HRTF for improved localisation in 3 D sound systems," Electronics letters 34.25, 1998.

[65] L. S. Davis, D. Ramani, G. Elena, A. G. Nail, L. Zhiyun and N. Z. Dmitry, "High order spatial audio capture and its binaural head-tracked playback over headphones with HRTF cues," *In Audio Engineering Society Convention 119. Audio Engineering Society,* 2005.

[66] D. H. Eberly, Game physics, CRC Press, 2010.

[67] I. Millington, Game Physics Engine Development, Amsterdam: Morgan Kaufmann Publishers, 2007.

[68] C. Pepper, S. Balakirsky and C. Scrapper, "Robot simulation physics validation," in *ACM: Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, 2007.

[69] F. Peinado and G. Pablo, "Transferring game mastering laws to interactive digital storytelling," in *Technologies for Interactive Digital Storytelling and Entertainment*, Berlin Heidelberg, Springer, 2004, pp. 48-54.

[70] G. David J., Introduction to Quantum Mechanics (2nd ed.), Prentice Hall, 2004.

[71] J. H. Haverkort, "Results on geometric networks and data structures," in *Dissertation*, Ytrecht University, 2004, pp. 9-10.

[72] A. Byagowi, D. Mohaddes and Z. Moussavi, "Design and Application of a Novel Virtual Reality Navigational Technology (VRNChair)," *Journal of experimental neuroscience 8,* no. 7, 2014.

[73] Drive Medical Design and Manufacturing, "https://www.drivemedical.com," 2012. [Online]. Available: https://www.drivemedical.com/catalog/product_info.php?cPath=87_144&products_id=978.

[74] R. K. Ahuja, K. Mehlhorn, J. Orlin and R. E. Tarjan, "Faster algorithms for the shortest path problem," *Jounal of ACM,* vol. 2, no. 37, pp. 213-223, 1990.

[75] K. Mehlhorn and P. Sanders, Algorithms and Data Structures: The Basic Toolbox, Springer, 2008.

[76] D. Zen, A. Byagowi, M. Garcia, D. Kelly, B. Lithgow and Z. Moussavi, "The perceived orientation in people with and without Alzheimer's," in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, San Diego, 2013.

[77] A. Byagowi, "Measuring Spatial Cognition Performance Using Virtual Reality," in *Design of Medical Devices*, Minneapolis, Minnesota, 2012.

[78] Logitech, "Logitech Attack™ 3 Joystick," 2012. [Online]. Available: http://www.logitech.com/en-ca/gaming/joysticks/302.

[79] W. Paul J., B. Ahmad and M. Zahra, "Effect of viewing mode on pathfinding in immersive Virtual Reality," *Engineering in Medicine and Biology Society (EMBC),* no. 37th Annual International Conference of the IEEE, pp. 4619-4622), 2015.

[80] A. Byagowi and Z. Moussavi, "Design of a Virtual Reality Navigational Experiment for Assessment of Egocentric Spatial Cognition," in *34th Annual International Conference of the IEEE EMBS*, San Diego, California, 2012.

[81] D. M. Johnson, "Introduction to and Review of Simulator Sickness Research," *US Armey Research Institute for the Behavioral and Social Science, Arlington,* 2005.

[82] Y. J. Kang, J. Ku, K. Han, S. I. Kim, T. W. Yu, J. H. Lee and C. I. Park, "Development and Clinical Trial of Virtual Reality-Based Cognitive Assessment in People with Stroke: Preliminary Study," *CyberPsychology and Behaviour,* vol. 11, no. 3, pp. 329-339, 2008.

[83] K. K. Zakzanis, G. Quintin, S. J. Graham and R. Mraz, "Age and dementia related differences in spatial navigation within an immersive virtual environment," *Medical Science Monitor,* vol. 15, no. 4, pp. 140-150, 2008.

[84] S. D. Moffat, A. D. Zonderman and S. M. Resnick, "Age Differences in Spatial Memory in a Virtual Environment Navigation Task," *Neurobiology of Aging,* vol. 22, no. 5, pp. 787-796, 2001.

[85] M. K. Rodgers, S. Joseph A. and M. Scott D., "Effects of Age on Navigation Strategy," *Neurobiology of Aging,* vol. 33, no. 1, pp. 202.e15-202.e22, 2012.

[86] A. Byagowi, "Man's Vision Based SLAM by Using Visual Odometry and Perspective Images," in *Fifth International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Linz, Austria, 2008.

[87] S. H. Stovall, Basic Inertial Navigation, China Lake, California, 1997.

[88] J. Lanier, "Virtually there," *Scientific American,* vol. 284, no. 4, pp. 66-75, 2001.

[89] S. Shannon, "The Chrome Age: Dawn of Virtual Reality," *Leonardo,* vol. 28, no. 5, pp. 369-380, 1995.

[90] I. Yavrucuk, E. Kubali and O. Tarimci, "A Low Cost Flight Simulator Using Virtual Reality Tools," *Aerospace and Electronic Systems Magazine, IEEE,* vol. 26, no. 4, pp. 10-14, 2011.

[91] J. C. Lee, "Hacking the Nintendo Wii Remote," *Pervasive Computing, IEEE,* vol. 7, no. 3, pp. 39 - 45, 2008.

[92] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *MultiMedia, IEEE,* vol. 19, no. 2, pp. 4-10, 2012.

[93] K. Sung, "Recent Videogame Console Technologis," *Computer, IEEE,* vol. 44, no. 2, pp. 91-93, 2011.

[94] C. Vapenstad and S. N. Buzink, "Procedural virtual reality simulation in minimally invasive surgery," *Surgical Endoscopy,* vol. 27, no. 2, pp. 364-377, 2013.

[95] T. P. Grantcharov, L. Bardram, P. Funch-Jensen and J. Rosenberg, "Impact of hand dominance, gender and experience with computer games on performance in virtual reality laparoscopy," *Surgical Endoscopy,* vol. 17, no. 7, pp. 1082-1085, 2003.

[96] D. Avola, M. Spezialetti and G. Placidi, "Design of an efficient framework for fast prototyping of customized human-computer interfaces and virtual environments for rehabilitation," *Computer Methods and Programs in Biomedicine,* vol. 110, no. 3, pp. 490-502, 2013.

[97] E. Hodgson and E. Bachmann, "Comparing Four Approaches to Generalized Redirected Walking: Simulation and Live User Data," in *Virtual Reality IEEE*, Orlando Florida, 2013.

[98] E. M. Kolasinski, "Simulator Sickness in Virtual Environments," U.S. Army Research Institute for the Behavioural and Social Sciences, Alexandria, Virginia, USA, 1995.

[99] J. Lanier, Interviewee, *Virtual reality: Meet founding father Jaron Lanier.* [Interview]. 19 June 2013.

[100] J. Brodkin, "How Unity3D Became a Game-Development Beast," DICE, 06 03 2013. [Online]. Available: http://insights.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/.

[101] L. Picurelli, "Yeeply," 15 09 2014. [Online]. Available: https://en.yeeply.com/blog/unity-3d-game-development-advantages-disadvantages/.

[102] D. Johnson, "Introduction to and Review of Simulator Sickness Research," U.S. Army Research Institute for the Behavioral and Social Sciences, Arlington, 2005.

[103] C. M. Armstrong, G. M. Reger, J. Edwards, A. A. Rizzo, C. G. Courtney and T. D. Parsons, "Validity of the Virtual Reality Stroop Rask (VRST) in active duty military," *Journal of Clinical and Experimental Neuropsychology,* vol. 35, no. 2, pp. 113-123, 2013.

[104] M. Denis and M. L. Jack, "Perspectives on human spatial cognition: memory, navigation, and environmental learning," *Psychological Research,* vol. 71.3, pp. 235-239, 2007.

[105] S. Gillner and A. M. Hanspeter, "Navigation and acquisition of spatial knowledge in a virtual maze," *Journal of Cognitive Neuroscience,* vol. 10.4, pp. 445-463, 1998.

[106] E. Hodgson, E. Bachmann, D. Waller, A. Bair and A. Oberlin, "Virtual Reality in the Wild: A Self-Contained and Wearable Simulation System," in *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, Costa Mesa, CA, 2012.

[107] Q. W. Ian, J. H. Dustin and G. W. Douglas, "Dead reckoning (path integration) requires the hippocampal formation: evidence from spontaneous exploration and spatial learning tasks in light (allothetic) and dark (idiothetic) tests," *Behavioural Brain Research 127,* pp. 49-69, 2001.

[108] R. Lord and W. John, The Theory of Sound 2Ed, MacMillan & Co, ed., 1896.

[109] Z. S. Nasreddine, A. P. Natalie, B. Valérie, C. Simon, W. Victor, C. Isabelle, L. C. Jeffrey and C. Howard, "The Montreal Cognitive Assessment (MoCA): A Brief Screening Tool For Mild Cognitive Impairment," *Journal of the American Geriatrics Society,* no. 53, pp. 695-699, 2005.

[110] T. D. Parsons, C. G. Courtney, M. E. Dawson, A. A. Rizzo and B. J. Arizmendi, "Visuospatial Processing and Learning Effects in Virtual Reality Based Mental Rotation and Navigational Tasks," *Engineering Psychology and Cognitive Ergonomics. Understanding Human Cognition. Lecture Notes in Computer Science,* vol. 8019, pp. 75-83, 2013.

[111] L. Pugnetti, L. Mendozzi, A. Motta, A. Cattaneo, E. Barbieri and A. Brancotti, "Evaluation and Retraining of Adults' Cognitive Impairments: Which Role for Virtual Reality Technology," *Computers in Biology and Medicine,* vol. 25, no. 2, pp. 213-227, 1995.

[112] N. E. Seymour, A. G. Gallagher, S. A. Roman, M. K. O'Brien, V. K. Bansal, D. K. Andersen and R. M. Satava, "Virtual Reality Training Improves Operating Room Performance," *Annals of Surgery,* vol. 236, no. 4, pp. 458-464, 2002.

[113] S. D. Moffat, "Age differences in spatial memory in a virtual environment navigation task," in *Neurobiol Aging*, Baltimore, Maryland, 2001.

[114] S. Ohyama, N. Suetaka, W. Hiroshi, M. Katsunori, A. Hironori, T. Noriaki and H. Tamotsu, "Autonomic responses during motion sickness induced by virtual reality," in *Auris Nasus Larynx*, Tokushima, Japan, 2007.

[115] C. Mueller, L. Michael, B. Sebastian, A. Daniela, L. Ralf, L. Michael and B. Johannes, "Building virtual reality fMRI paradigms: A framework for presenting immersive virtual environments," in *J Neurosci Methods*, Magdeburg, Germany, 2012.

[116] C. Massot, S. Adam D., C. Maurice J. and C. Kathleen E., "The vestibular system implements a linear-nonlinear transformation in order to encode self-motion," in *PLoS Biol.*, Montreal, Quebec, Canada, 2012.

[117] H. Sauzéon, A. P. Prashant, L. Florian, W. Gregory, D. Marie, Z. Xia, G. Pascal and N. Bernard, "The use of virtual reality for episodic memory assessment: effects of active navigation," *Exp Psychol.,* vol. 59, no. II, pp. 99-108, 2011.

[118] D. Peterson and C. Robertson, "Utilizing Virtual Reality in Teaching and Training: Progress Toward Virtual Surgical Simulations," in *7th International Technology Education and Development Conference*, Valencia, Spain, 2013.

Appendix A

Matlab Implementation of the Spatial Resampling Algorithm

Start of “EquiDistanceCenversion.m” ----------

```
function equiDist = EquiDistanceConversion(inputTraj,min_dis)
%EQUIDISTANCE compute the equidistant transform for a trajectory
%   compute a new trajectory from INPUTTRAJ, such that in the new
%   trajectory, all points are equidistant with distance MIN_DIS.
%
%   INPUTTRAJ a 3-column matrix, where each column corresponds to a
%   dimension of the trajectory in space.
%
%   MIN_DIS a double representing the desired spatial resolution (distance
%   between each point)

inputTraj_size = length(inputTraj);
equiDist(1,:) = inputTraj(1,:);
output_index = 1;
input_index = 1;

%figure(5); hold; figure(6); hold;
while (input_index < inputTraj_size)
   sphere = [equiDist(output_index, :) min_dis]; % a sphere centred at the last point added, with radius
min_dis
   point = inputTraj(input_index, :); % current point
   nextPoint = inputTraj(input_index + 1, :); % next point
   if ~InSphere(nextPoint, sphere) % && (InSphere(point, sphere)
      % compute intersection
      line = [point nextPoint];
      Intersection_point = intersectLineSphere(line, sphere);
      output_index = output_index + 1;
      equiDist(output_index,:) = Intersection_point;
   elseif InSphere(nextPoint, sphere)
      input_index = input_index + 1;
   end
end
end
```

End of “EquiDistanceCenversion.m” ----------

Start of “InSphere.m” ----------

```
function [ inSphere ] = InSphere( point, sphere )
%INSPHERE determine if POINT is in SPHERE
%
%   POINT is a 1x3 row-vector, where the first 3 indices represent the x,
%   y, and z coordinates respectively.
%
%   SPHERE is a 1x4 row-vector, where the first 3 indices represent the
%   location of the sphere in space, and the 4th index represents the
%   sphere's radius.
%
%   INSPHERE is a logical value indicating whether or not POINT is in
%   SPHERE

% transform POINT so that it is relative to SPHERE
radius = sphere(1, 4);
point_rel_sphere = point - sphere(1, (1:3));
inSphere = sqrt(point_rel_sphere(1, 1)^2 +  point_rel_sphere(1, 2)^2 + point_rel_sphere(1, 3)^2)
<= radius;

end
```

End of “InSphere.m” ----------

Start of “IntersectLineSphere.m” ----------

```
function point = IntersectLineSphere(line, sphere)
%INTERSECTLINESPHERE identify the point at which LINE intersects SPHERE
%
%   LINE is a 1x6 row-vector, where the first 3 indices represent the start
%   point of the line and the last 3 indices represent the end point.
%
%   SPHERE is a 1x4 row-vector, where the first 3 indices represent the
%   location of the sphere in space, and the 4th index represents the
%   sphere's radius.
%
%   POINT is the 1x3 row vector that contains the x, y, z coordinates of
%   the intersection point in space.

r = sphere(1,4);
p2 = line(4:6);
p1 = line(1:3);

% x_c, y_c, z_c represent the centre of the sphere
x_c = sphere(1, 1);
y_c = sphere(1, 2);
z_c = sphere(1, 3);
```

```
% x_0, y_0, z_0 represent the starting point of the line segment (which is
% inside the sphere)
x_0 = p1(1, 1);
y_0 = p1(1, 2);
z_0 = p1(1, 3);

% x_1, y_1, z_1 represent the endpoint of the line segment (which is
% outside the sphere)
x_1 = p2(1, 1);
y_1 = p2(1, 2);
z_1 = p2(1, 3);


% Using the formula from http://stackoverflow.com/questions/5883169/intersection-between-a-line-and-a-sphere
A = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - r^2;
C = (x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2;
B = (x_1 - x_c)^2 + (y_1 - y_c)^2 + (z_1 - z_c)^2 - A - C - r^2;
% Solve A + B*t + C*t^2 = 0 for t. This is a normal quadratic equation.
t_solutions = roots([C B A]);

% the desired value of t will be between 0 and 1
if size(t_solutions, 1) < 1
    error('Line does not intersect sphere');
elseif size(t_solutions, 1) == 1
    t = t_solutions(1, 1);
elseif size(t_solutions, 1) == 2
    if t_solutions(1, 1) > 0
        t = t_solutions(1, 1);
    else
        t = t_solutions(2, 1);
    end
end

% the target point is partway between p1 and p2
point = p1 + t * (p2 - p1);
end
```

End of “IntersectLineSphere.m” ----------

Appendix B

Parts of *Virtual House* C++ code

Start of “main.cpp” ----------

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "OVR.h"
#include "generic.h"
#include "settings.h"
#include "projectsetup.h"
#include "projectloop.h"
#include "projectshaders.h"

#pragma comment(lib,"libovr.lib")

int main(int argc,char **argv){

	for(int i=1;i<argc;i++){
	  if(!strcmp(argv[i],"/TEST")) test=atoi(argv[i+1]);
	}
	System::Init(Log::ConfigureDefaultLog(LogMask_All));

	load_OVR();

	loadgenericsettings();                                          //load
the generic settings


	genericsetup();
	//generic setup

	dashsetup();
	//setup the dashboard

	projectsetup();
	//setup for the project

	updateinput();                                                  //get
the initial value of the HMD
	Start_yaw=camyang-(90*radiansindegree);

	initShaders(file2string("k.vert"), file2string("k.frag"));
	initFBO(screenw, screenh);

	//game loop
	while(!shutdownprogram)
	{
```

```
		glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);  //clear the screen

		SDL_PumpEvents();                                                    //get
what events have occured

		updateinput();                                                       //get
controller input

		gamespeed=60.f/dash_framerate;
		if(gamespeed<0.2f)gamespeed=0.2f;
		if(gamespeed>2.0f)gamespeed=2.0f;

		//play
		play(gamespeed);

		if(!dashonpercent) RenderScene();

		dashloop();

		SDL_GL_SwapBuffers();

		//clear out left over events and shut down when appropriate
		SDL_Event event;
		while(SDL_PeepEvents(&event,1,SDL_GETEVENT,SDL_ALLEVENTS)>0)
			if(event.type==SDL_QUIT)shutdownprogram=1;
	}

	// Clean up
	SKYBOX_Finalize();
	SDL_Quit();
	return 0;
}
```

End of “main.cpp” ----------

Start of “drawworld.h” ----------

```
GLuint vertexbuffer;
GLuint colorbuffer;

void setupdraw()
{
	glGenBuffers(1, &vertexbuffer);
	glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
	glBufferData(GL_ARRAY_BUFFER, sizeof(g_vertex_buffer_data), g_vertex_buffer_data,
GL_STATIC_DRAW);

	glGenBuffers(1, &colorbuffer);
```

```
	glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
	glBufferData(GL_ARRAY_BUFFER, sizeof(g_color_buffer_data), g_color_buffer_data,
GL_STATIC_DRAW);
}

extern double dyndist;
void drawworld(Eye LorR){
	//glUseProgram(program_object);
	//set the draw distance for clipping and fog


	float camnear=0.1f;
	float camfar=worldtilesize*worldtileviewrange;

	//get cam view range
	int lowxview=highint(0,playerxgridpos-worldtileviewrange);
	int highxview=lowint(worldgridsizex,playerxgridpos+worldtileviewrange+1);
	int lowyview=highint(0,playerygridpos-worldtileviewrange);
	int highyview=lowint(worldgridsizey,playerygridpos+worldtileviewrange+1);
	int lowzview=highint(0,playerzgridpos-worldtileviewrange);
	int highzview=lowint(worldgridsizez,playerzgridpos+worldtileviewrange+1);

	//setup the camera
	//set3dcamera(camxpos,camypos,camzpos,-
camxang,camyang+90*radiansindegree,camzang,45,4/3); //screena);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();

	//set3dcamera(playerxpos, playerypos, playerzpos, 0, nav_camyang, 0, 45, 4/3);
	float eyedist=dyndist/2;
	if(LorR==Left)
	  set3dcamera(camxpos-(cos(camyang)*eyedist),camypos,camzpos-
(sin(camyang)*eyedist),camxang,camyang,camzang,90,1);
	else

	set3dcamera(camxpos+(cos(camyang)*eyedist),camypos,camzpos+(sin(camyang)*eyedis
t),camxang,camyang,camzang,90,1);
	//prep for drawing
	if(usetextures==1)glEnable(GL_TEXTURE_2D);
	glEnableClientState(GL_VERTEX_ARRAY);
	glEnableClientState(GL_TEXTURE_COORD_ARRAY);

	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	GLfloat light_position0[] = {camxpos,camypos,camzpos,1.0};
	GLfloat light_ambient0[] = {1.0,1.0,1.0,1.0};
	GLfloat light_diffuse0[] = {1.0,1.0,1.0,1.0};
	GLfloat light_specular0[] = {1.0,1.0,1.0,1.0};
	glLightfv(GL_LIGHT0,GL_POSITION,light_position0);
	glLightfv(GL_LIGHT0,GL_AMBIENT,light_ambient0);
	glLightfv(GL_LIGHT0,GL_DIFFUSE,light_diffuse0);
```

```
        glLightfv(GL_LIGHT0,GL_SPECULAR,light_specular0);
//      glLightf(GL_LIGHT0,GL_QUADRATIC_ATTENUATION,0.005f);
        glEnableClientState(GL_NORMAL_ARRAY);

        //draw the world tiles
        for(int x=lowxview; x<highxview; x++)
          for(int y=lowyview; y<highyview; y++)
                for(int z=lowzview; z<highzview; z++){
                        int b=worldgrid[x][y][z][0];
                        if(b>0){
                                int c=worldgrid[x][y][z][1];
                                glPushMatrix();
                                glBindTexture(GL_TEXTURE_2D,worldtiletexture[b]);
                                glTranslatef(x*worldtilesize,y*worldtilesize,z*worldtilesize);
                                draw3dtrianglemesh(worldtilevertexcount[b],
        worldtilevertexarray[b][c],worldtiletexturearray[b],NULL,worldtilenormalarray[b][c]);
                                glPopMatrix();
                        }
                }


  if(usetextures)glBindTexture(GL_TEXTURE_2D,wireframetexture);
  if(usetextures){
          glEnable(GL_TEXTURE_2D);
          glEnableClientState(GL_TEXTURE_COORD_ARRAY);
          glBindTexture(GL_TEXTURE_2D,entitytexture[1]);
        }
   for(int i=0;i<maxdoors;i++){
          glEnable(GL_TEXTURE_2D);
          glEnableClientState(GL_TEXTURE_COORD_ARRAY);
          glBindTexture(GL_TEXTURE_2D,entitytexture[entitytype_door]);
          glPushMatrix();
 if (i>=0 && i<4){
        switch(door_array[i][3]){
                case 2:
                        glTranslatef(door[i].xpos-4.25,door[i].ypos-4,door[i].zpos-2.75);
//4.25          //glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
              glRotatef(door[i].yang+180,0,1,0);
                        glTranslatef(0,0,-2.45);
                break;
                case 1:
                        glTranslatef(door[i].xpos-2.35,door[i].ypos-4,door[i].zpos-4);
//glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang+270,0,1,0);
                        glTranslatef(0,0,-2);
                        break;
                case 0:
```

```
                        glTranslatef(door[i].xpos+3.5,door[i].ypos-4,door[i].zpos+2.85);
//glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang,0,1,0);
                        glTranslatef(0,0,-2.45);
                        break;
                case 3:
                        glTranslatef(door[i].xpos-2.5,door[3].ypos-4,door[3].zpos+4);
//glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang-90,0,1,0);
                        glTranslatef(0,0,-2);
                        break;
        }
 }
    else if(i>=4)
{
        switch(door_array[i][3]){
                case 2:
                        glTranslatef(door[i].xpos+4.25,door[i].ypos-4,door[i].zpos-2.40);
//4.25          //glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang+180,0,1,0);
                        glTranslatef(0,0,-2);
                break;
                case 1:
                        glTranslatef(door[i].xpos-2.45,door[i].ypos-4,door[i].zpos+4);
//glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang+270,0,1,0);
                        glTranslatef(0,0,-2);
                        break;
                case 0:
                        glTranslatef(door[i].xpos-4.25,door[i].ypos-4,door[i].zpos+2.40);
//glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang,0,1,0);
                        glTranslatef(0,0,-2);
                        break;
                case 3:
                        glTranslatef(door[i].xpos+2.45,door[i].ypos-4,door[i].zpos-4);
//glTranslatef(ex_x,ex_y,ex_z);
                        glRotatef(door[i].xang,1,0,0);
                        glRotatef(door[i].zang,0,0,1);
                        glRotatef(door[i].yang+90,0,1,0);
                        glTranslatef(0,0,-2);
```

```
                        break;
        }
}


glVertexPointer(3,GL_FLOAT,sizeof(InterleavedVertex),&entityinterleavedvertex[door[i].type]->px);

        glTexCoordPointer(2,GL_FLOAT,sizeof(InterleavedVertex),&entityinterleavedvertex[door[i].type]->tx);

        glNormalPointer(GL_FLOAT,sizeof(InterleavedVertex),&entityinterleavedvertex[door[i].type]->nx);
          glDrawArrays(GL_TRIANGLES,0,entityvertexcount[door[i].type]);
          glPopMatrix();
  }
        //done drawing
        glDisableClientState(GL_VERTEX_ARRAY);
        glDisableClientState(GL_TEXTURE_COORD_ARRAY);
        glDisable(GL_TEXTURE_2D);

        //turn off lighting
        glDisable(GL_LIGHTING);
        glDisable(GL_LIGHT0);

        //turn off fog
        glDisable(GL_FOG);

}
```

End of “drawworld.h” ----------

Start of “genworldgrid.h” ----------

```
const int streetlevel=3;
const int minbuildingheight=3;
const int maxbuildingheight=9;

const int worldgridsizex = 50;
const int worldgridsizey = 15;
const int worldgridsizez = 50;

//street width setting
const int streetwidth  = 2;

#define GRASS_TILE                      10
#define SIDEWALK_CORNER_TILE9
#define SIDEWALK_TILE                   8
const int clue_win_array[9][9]={
                {000,00,0,00,00,0,00,00,00},
                {'M',18,4,21,18,4,28,27,25},                    //1
```

```
                {'D',21,5,31,28,5,31,27,31},                    //2
                {'J',28,5,18,21,5,18,28,33},                    //3
                {'S',31,4,28,31,4,21,26,28},                    //4
                {'V',28,4,18,21,4,18,26,27},                    //5
                {'G',31,5,28,31,5,21,29,27},                    //6
                {'P',21,4,31,28,4,31,25,25},                    //7
                {'A',18,5,21,18,5,28,29,26}};

char worldgrid[worldgridsizex][worldgridsizey][worldgridsizez][2];

//clean up
void genworldgrid_cleanup(){

        //clean the world grid
        for(int x=0; x<worldgridsizex; x++)
           for(int y=0; y<worldgridsizey; y++)
                   for(int z=0; z<worldgridsizez; z++){
                           worldgrid[x][y][z][0]=0;
                           worldgrid[x][y][z][1]=0;
                   }

}

//island streets and sidewalks
void genworldgrid_street(){

        //mid point position
        int worldmidposx = worldgridsizex / 2;
        int worldmidposz = worldgridsizez / 2;

        for(int x=0; x<worldgridsizex; x++)
           for(int z=0; z<worldgridsizez; z++)
                   worldgrid[x][streetlevel][z][0]=GRASS_TILE;

        for(int x=0; x<worldgridsizex; x++)
           for(int z=0; z<worldgridsizez; z++)
                   worldgrid[x][streetlevel][z][0]=GRASS_TILE;



}

//Foundation of the building
void genworldgrid_sidewalk(){

        //sidewalks
        for(int z=house_low_z+1; z<=(house_high_z-1); z++){
           worldgrid[house_low_x][streetlevel][z][0]=8;
           worldgrid[house_low_x][streetlevel][z][1]=0;
           worldgrid[house_high_x][streetlevel][z][0]=8;
           worldgrid[house_high_x][streetlevel][z][1]=2;
```

```
        }
        for(int x=house_low_x+1; x<=(house_high_x-1); x++){
           worldgrid[x][streetlevel][house_low_z][0]=8;
           worldgrid[x][streetlevel][house_low_z][1]=3;
           worldgrid[x][streetlevel][house_high_z][0]=8;
           worldgrid[x][streetlevel][house_high_z][1]=1;
        }

        //sidewalk corners
        worldgrid[house_high_z][streetlevel][house_low_z][0]=9;
        worldgrid[house_high_z][streetlevel][house_low_z][1]=3;
        worldgrid[house_low_z][streetlevel][house_low_z][0]=9;
        worldgrid[house_low_z][streetlevel][house_low_z][1]=0;
        worldgrid[house_low_z][streetlevel][house_high_z][0]=9;
        worldgrid[house_low_z][streetlevel][house_high_z][1]=1;
        worldgrid[house_high_z][streetlevel][house_high_z][0]=9;
        worldgrid[house_high_z][streetlevel][house_high_z][1]=2;
}

//buildings
void genworldgrid_building(){


        //wall
        for (int y=0;y<1;y++)
        {
           for(int z=house_low_z+2; z<=(house_high_z-2); z++){
                    worldgrid[house_low_x+1][streetlevel+y][z][0]=72;
                    worldgrid[house_low_x+1][streetlevel+y][z][1]=0;
                    worldgrid[house_high_x-1][streetlevel+y][z][0]=72;
                    worldgrid[house_high_x-1][streetlevel+y][z][1]=2;
           }
           for(int x=house_low_x+2; x<=(house_high_x-2); x++){
                    worldgrid[x][streetlevel+y][house_low_z+1][0]=72;
                    worldgrid[x][streetlevel+y][house_low_z+1][1]=3;
                    worldgrid[x][streetlevel+y][house_high_z-1][0]=72;
                    worldgrid[x][streetlevel+y][house_high_z-1][1]=1;
           }

           worldgrid[house_high_z-1][streetlevel+y][house_low_z+1][0]=71;
           worldgrid[house_high_z-1][streetlevel+y][house_low_z+1][1]=2;
           worldgrid[house_low_z+1][streetlevel+y][house_low_z+1][0]=71;
           worldgrid[house_low_z+1][streetlevel+y][house_low_z+1][1]=3;
           worldgrid[house_low_z+1][streetlevel+y][house_high_z-1][0]=71;
           worldgrid[house_low_z+1][streetlevel+y][house_high_z-1][1]=0;
           worldgrid[house_high_z-1][streetlevel+y][house_high_z-1][0]=71;
           worldgrid[house_high_z-1][streetlevel+y][house_high_z-1][1]=1;
        }

        //building windows
        //wall 1
```

```
        worldgrid[house_low_x+4][streetlevel][house_low_z+1][0]=window_tile;
        worldgrid[house_low_x+4][streetlevel][house_low_z+1][1]=3;
        worldgrid[house_low_x+7][streetlevel][house_low_z+1][0]=window_tile;
        worldgrid[house_low_x+7][streetlevel][house_low_z+1][1]=3;
        worldgrid[house_low_x+11][streetlevel][house_low_z+1][0]=window_tile;
        worldgrid[house_low_x+11][streetlevel][house_low_z+1][1]=3;
        //wall 2
        worldgrid[house_low_x+1][streetlevel][house_high_z-4][0]=window_tile;
        worldgrid[house_low_x+1][streetlevel][house_high_z-4][1]=0;
        worldgrid[house_low_x+1][streetlevel][house_high_z-7][0]=window_tile;
        worldgrid[house_low_x+1][streetlevel][house_high_z-7][1]=0;
        worldgrid[house_low_x+1][streetlevel][house_high_z-11][0]=window_tile;
        worldgrid[house_low_x+1][streetlevel][house_high_z-11][1]=0;
        //wall 3
        worldgrid[house_low_x+4][streetlevel][house_high_z-1][0]=window_tile;
        worldgrid[house_low_x+4][streetlevel][house_high_z-1][1]=1;
        worldgrid[house_low_x+8][streetlevel][house_high_z-1][0]=window_tile;
        worldgrid[house_low_x+8][streetlevel][house_high_z-1][1]=1;
        worldgrid[house_low_x+11][streetlevel][house_high_z-1][0]=window_tile;
        worldgrid[house_low_x+11][streetlevel][house_high_z-1][1]=1;
        //wall 4
        worldgrid[house_high_x-1][streetlevel][house_high_z-4][0]=window_tile;
        worldgrid[house_high_x-1][streetlevel][house_high_z-4][1]=2;
        worldgrid[house_high_x-1][streetlevel][house_high_z-8][0]=window_tile;
        worldgrid[house_high_x-1][streetlevel][house_high_z-8][1]=2;
        worldgrid[house_high_x-1][streetlevel][house_high_z-11][0]=window_tile;
        worldgrid[house_high_x-1][streetlevel][house_high_z-11][1]=2;


        expectdist=0;    //reset the expected distance value



        //building door
        worldgrid[house_low_x+8][streetlevel][house_low_z+1][0]=door_tile;
        worldgrid[house_low_x+7][streetlevel][house_high_z-1][0]=door_tile;
        int entrydoorz=house_high_z-3-(rand()%(house_high_z-house_low_z-6));
        worldgrid[house_low_x+1][streetlevel][house_low_z+7][0]=door_tile;
        worldgrid[house_high_x-1][streetlevel][house_low_z+8][0]=door_tile;

        //calculated the expected distance
        //to the entrance door
        expectdist += sqrt(((float)((worldgridsizex/2)-((worldgridsizex/3))-
(house_low_x+1))*((worldgridsizex/2)-((worldgridsizex/3))-(house_low_x+1)))
                                +(((worldgridsizex/2)-(entrydoorz))*((worldgridsizex/2)-
(entrydoorz))));
        //to the stairs
        expectdist += sqrt((((23.6)-(house_low_x+1))*((23.6)-(house_low_x+1)))
                                +(((23.3)-(entrydoorz))*((23.3)-(entrydoorz))));

        //floor base
```

```
        for(int y=0;y<=2;y++)
           for(int z=house_low_z+2; z<=(house_high_z-2); z++)
                    for(int x=house_low_x+2; x<=(house_high_x-2); x++)
                             worldgrid[x][streetlevel+y][z][0]=hardwood_tile;//stairtype

        switch(stairtype){
        case 1:
//stairs at 180 degrees
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+1][house_low_z+((house_high_z-
house_low_z)/2)][0]=stairs_tile;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+2][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+2][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)][0]=stairs_tile;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][0]=stairs_tile;
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][1]=2;
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+2][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+2][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-
house_low_z)/2)+1][0]=stairs_tile;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][1]=2;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
        break;

        case 2:
        //stairs at 90 degrees (floor not fixed for this configuration)
        //holes in floor
```

```
        //second floor floor
        worldgrid[25][streetlevel+1][24][0]=empty_tile;//top of stairs...
        worldgrid[25][streetlevel+1][25][0]=empty_tile;
        worldgrid[24][streetlevel+1][24][0]=empty_tile;
        worldgrid[24][streetlevel+1][25][0]=empty_tile;//...
        worldgrid[26][streetlevel+1][23][0]=empty_tile;//bottom of stairs...
        worldgrid[27][streetlevel+1][23][0]=empty_tile;
        worldgrid[23][streetlevel+1][26][0]=empty_tile;
        worldgrid[22][streetlevel+1][26][0]=empty_tile;//...
        //3rd floor floor
        worldgrid[26][streetlevel+2][23][0]=empty_tile;
        worldgrid[27][streetlevel+2][23][0]=empty_tile;
        worldgrid[23][streetlevel+2][26][0]=empty_tile;
        worldgrid[22][streetlevel+2][26][0]=empty_tile;
        //isolators

        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-
house_low_z)/2)+1][0]=stairs_tile;//first floor up
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-house_low_z)/2)][0]=0;//bottom
of first floor stairs
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][1]=2;//first
floor up rotation
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-
house_low_z)/2)][0]=stairs_tile;//first floor up
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;//bottom
of first floor stairs
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)-
1][0]=stairs2_tile;//second floor up//was plus 1 z
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)-
1][1]=1;//was2;//second floor up rotation
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+2][0]=stairs2_tile;//second floor
up//was plus 1 z
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+2][1]=3;//was2;//second floor
up rotation
        break;
        case 3:

        //ramps at 180 degrees
    worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel+1][house_low_z+((house_high_z-
house_low_z)/2)][0]=ramp_tile;
        worldgrid[house_low_x+((house_high_x-
```

```
house_low_x)/2)+1][streetlevel+2][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel+2][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][0]=ramp_tile;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][1]=2;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)][0]=0;
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)][0]=0;

	break;

	case 4:
	//holes in floor
	//second floor floor
	worldgrid[25][streetlevel+1][24][0]=empty_tile;//top of stairs...
	worldgrid[25][streetlevel+1][25][0]=empty_tile;
	worldgrid[24][streetlevel+1][24][0]=empty_tile;
	worldgrid[24][streetlevel+1][25][0]=empty_tile;//...
	worldgrid[26][streetlevel+1][23][0]=empty_tile;//bottom of stairs...
	worldgrid[27][streetlevel+1][23][0]=empty_tile;
	worldgrid[23][streetlevel+1][26][0]=empty_tile;
	worldgrid[22][streetlevel+1][26][0]=empty_tile;//...
	//3rd floor floor
	worldgrid[26][streetlevel+2][23][0]=empty_tile;
	worldgrid[27][streetlevel+2][23][0]=empty_tile;
	worldgrid[23][streetlevel+2][26][0]=empty_tile;
	worldgrid[22][streetlevel+2][26][0]=empty_tile;

	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-
house_low_z)/2)+1][0]=ramp_tile;//first floor up
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-house_low_z)/2)][0]=0;//bottom
of first floor stairs
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+1][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][1]=2;//first
floor up rotation
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)][0]=ramp_tile;//first
floor up
	worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel][house_low_z+((house_high_z-house_low_z)/2)+1][0]=0;//bottom
of first floor stairs
	worldgrid[house_low_x+((house_high_x-
```

```
house_low_x)/2)+2][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)-
1][0]=ramp2_tile;//second floor up//was plus 1 z
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)+2][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)-
1][1]=1;//was2;//second floor up rotation
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+2][0]=ramp2_tile;//second floor
up//was plus 1 z
        worldgrid[house_low_x+((house_high_x-house_low_x)/2)-
1][streetlevel+1][house_low_z+((house_high_z-house_low_z)/2)+2][1]=3;//was2;//second floor
up rotation
        break;
        }
        //roof
        worldgrid[house_low_x+((house_high_x-
house_low_x)/2)][streetlevel+3][house_low_z+((house_high_z-house_low_z)/2)][0]=roof_tile;

        if(allocentric)
        {
          worldgrid[23][3][30][0]=timhortonswall_tile;  ////////////////////////////
          worldgrid[26][4][30][0]=atmmachinewall_tile;
        }

for(int y=streetlevel+1;y<=streetlevel+2;y++){
          worldgrid[house_low_x+7][y][house_low_z+2][0]=interior_wallcorner_tile_l;///done
          worldgrid[house_low_x+7][y][house_low_z+2][1]=0;
          worldgrid[house_low_x+8][y][house_low_z+2][0]=interior_wallcorner_tile_l;
          worldgrid[house_low_x+8][y][house_low_z+2][1]=0;
          worldgrid[house_low_x+2][y][house_high_z-7][0]=interior_wallcorner_tile_l;
          worldgrid[house_low_x+2][y][house_high_z-7][1]=1;
          worldgrid[house_low_x+2][y][house_high_z-8][0]=interior_wallcorner_tile_l;
          worldgrid[house_low_x+2][y][house_high_z-8][1]=1;
          worldgrid[house_low_x+7][y][house_high_z-2][0]=interior_wallcorner_tile_l;
          worldgrid[house_low_x+7][y][house_high_z-2][1]=2;
          worldgrid[house_low_x+8][y][house_high_z-2][0]=interior_wallcorner_tile_l;
          worldgrid[house_low_x+8][y][house_high_z-2][1]=2;
          worldgrid[house_high_x-2][y][house_high_z-8][0]=interior_wallcorner_tile_l;
          worldgrid[house_high_x-2][y][house_high_z-8][1]=3;
          worldgrid[house_high_x-2][y][house_high_z-7][0]=interior_wallcorner_tile_l;
          worldgrid[house_high_x-2][y][house_high_z-7][1]=3;
          //interior_wallcorner_tile is important to block the middle rums
          //walls
          //1 outside don't want it
          worldgrid[house_low_x+1][y][house_low_z+1][0]=wallcorner_tile;
          worldgrid[house_low_x+1][y][house_low_z+1][1]=3;
          worldgrid[house_low_x+2][y][house_low_z+1][0]=wall_tile;
          worldgrid[house_low_x+2][y][house_low_z+1][1]=3;
          worldgrid[house_low_x+3][y][house_low_z+1][0]=inout_wallcorner_tile_l;
          worldgrid[house_low_x+3][y][house_low_z+1][1]=2;
          worldgrid[house_low_x+4][y][house_low_z+1][0]=window_tile;
          worldgrid[house_low_x+4][y][house_low_z+1][1]=3;
```

```
        worldgrid[house_low_x+5][y][house_low_z+1][0]=inout_wallcorner_tile_r;
        worldgrid[house_low_x+5][y][house_low_z+1][1]=2;
        worldgrid[house_low_x+6][y][house_low_z+1][0]=inout_wallcorner_tile_l;
        worldgrid[house_low_x+6][y][house_low_z+1][1]=2;
        worldgrid[house_low_x+7][y][house_low_z+1][0]=double_window_tile;
        worldgrid[house_low_x+7][y][house_low_z+1][1]=1;
        worldgrid[house_low_x+9][y][house_low_z+1][0]=inout_wallcorner_tile_r;
        worldgrid[house_low_x+9][y][house_low_z+1][1]=2;
        worldgrid[house_low_x+10][y][house_low_z+1][0]=inout_wallcorner_tile_l;
        worldgrid[house_low_x+10][y][house_low_z+1][1]=2;
        worldgrid[house_low_x+11][y][house_low_z+1][0]=window_tile;
        worldgrid[house_low_x+11][y][house_low_z+1][1]=3;
        worldgrid[house_low_x+12][y][house_low_z+1][0]=inout_wallcorner_tile_r;
        worldgrid[house_low_x+12][y][house_low_z+1][1]=2;
        worldgrid[house_low_x+13][y][house_low_z+1][0]=wall_tile;
        worldgrid[house_low_x+13][y][house_low_z+1][1]=3;
        //1 inside
        worldgrid[house_low_x+1][y][house_low_z+2][0]=wall_tile;
        worldgrid[house_low_x+1][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+2][y][house_low_z+2][0]=block_tile;
        worldgrid[house_low_x+2][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+3][y][house_low_z+2][0]=interior_wallcorner_tile_l;
        worldgrid[house_low_x+3][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+4][y][house_low_z+2][0]=door_tile;
        worldgrid[house_low_x+4][y][house_low_z+2][1]=1;
        worldgrid[house_low_x+5][y][house_low_z+2][0]=interior_wallcorner_tile_r;
        worldgrid[house_low_x+5][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+6][y][house_low_z+2][0]=interior_wallcorner_tile_l;
        worldgrid[house_low_x+6][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+9][y][house_low_z+2][0]=interior_wallcorner_tile_r;
        worldgrid[house_low_x+9][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+10][y][house_low_z+2][0]=interior_wallcorner_tile_l;
        worldgrid[house_low_x+10][y][house_low_z+2][1]=0;
        worldgrid[house_low_x+11][y][house_low_z+2][0]=door_tile;
        worldgrid[house_low_x+11][y][house_low_z+2][1]=1;
        worldgrid[house_low_x+12][y][house_low_z+2][0]=interior_wallcorner_tile_r;
        worldgrid[house_low_x+12][y][house_low_z+2][1]=0;
//      interior_wallcorner_tile is important for saloni it changes internal things
        //2 outside
        worldgrid[house_low_x+1][y][house_high_z-1][0]=wallcorner_tile;
        worldgrid[house_low_x+1][y][house_high_z-1][1]=0;
        worldgrid[house_low_x+1][y][house_high_z-2][0]=wall_tile;
        worldgrid[house_low_x+1][y][house_high_z-2][1]=0;
        worldgrid[house_low_x+1][y][house_high_z-3][0]=inout_wallcorner_tile_l;
        worldgrid[house_low_x+1][y][house_high_z-3][1]=3;
        worldgrid[house_low_x+1][y][house_high_z-4][0]=window_tile;
        worldgrid[house_low_x+1][y][house_high_z-4][1]=0;
        worldgrid[house_low_x+1][y][house_high_z-5][0]=inout_wallcorner_tile_r;
        worldgrid[house_low_x+1][y][house_high_z-5][1]=3;
        worldgrid[house_low_x+1][y][house_high_z-6][0]=inout_wallcorner_tile_l;
        worldgrid[house_low_x+1][y][house_high_z-6][1]=3;
```

```
 worldgrid[house_low_x+1][y][house_high_z-7][0]=double_window_tile;
worldgrid[house_low_x+1][y][house_high_z-7][1]=2;
worldgrid[house_low_x+1][y][house_high_z-9][0]=inout_wallcorner_tile_r;
worldgrid[house_low_x+1][y][house_high_z-9][1]=3;
worldgrid[house_low_x+1][y][house_high_z-10][0]=inout_wallcorner_tile_l;
worldgrid[house_low_x+1][y][house_high_z-10][1]=3;
worldgrid[house_low_x+1][y][house_high_z-11][0]=window_tile;
worldgrid[house_low_x+1][y][house_high_z-11][1]=0;
worldgrid[house_low_x+1][y][house_high_z-12][0]=inout_wallcorner_tile_r;
worldgrid[house_low_x+1][y][house_high_z-12][1]=3;
//2 inside front side
worldgrid[house_low_x+2][y][house_high_z-3][0]=interior_wallcorner_tile_l;
worldgrid[house_low_x+2][y][house_high_z-3][1]=1;
worldgrid[house_low_x+2][y][house_high_z-4][0]=door_tile;
worldgrid[house_low_x+2][y][house_high_z-4][1]=2;
worldgrid[house_low_x+2][y][house_high_z-5][0]=interior_wallcorner_tile_r;
worldgrid[house_low_x+2][y][house_high_z-5][1]=1;
worldgrid[house_low_x+2][y][house_high_z-6][0]=interior_wallcorner_tile_l;
worldgrid[house_low_x+2][y][house_high_z-6][1]=1;
worldgrid[house_low_x+2][y][house_high_z-9][0]=interior_wallcorner_tile_r;
worldgrid[house_low_x+2][y][house_high_z-9][1]=1;
worldgrid[house_low_x+2][y][house_high_z-10][0]=interior_wallcorner_tile_l;
worldgrid[house_low_x+2][y][house_high_z-10][1]=1;
worldgrid[house_low_x+2][y][house_high_z-11][0]=door_tile;
worldgrid[house_low_x+2][y][house_high_z-11][1]=2;
worldgrid[house_low_x+2][y][house_high_z-12][0]=interior_wallcorner_tile_r;
worldgrid[house_low_x+2][y][house_high_z-12][1]=1;


// 3 outside
 worldgrid[house_low_x+1][y][house_high_z-1][0]=wallcorner_tile;
worldgrid[house_low_x+1][y][house_high_z-1][1]=0;
worldgrid[house_low_x+2][y][house_high_z-1][0]=wall_tile;
worldgrid[house_low_x+2][y][house_high_z-1][1]=1;
worldgrid[house_low_x+3][y][house_high_z-1][0]=inout_wallcorner_tile_r;
worldgrid[house_low_x+3][y][house_high_z-1][1]=0;
worldgrid[house_low_x+4][y][house_high_z-1][0]=window_tile;
worldgrid[house_low_x+4][y][house_high_z-1][1]=1;
worldgrid[house_low_x+5][y][house_high_z-1][0]=inout_wallcorner_tile_l;
worldgrid[house_low_x+5][y][house_high_z-1][1]=0;
worldgrid[house_low_x+6][y][house_high_z-1][0]=inout_wallcorner_tile_r;
worldgrid[house_low_x+6][y][house_high_z-1][1]=0;
 worldgrid[house_low_x+8][y][house_high_z-1][0]=double_window_tile;
worldgrid[house_low_x+8][y][house_high_z-1][1]=3;
worldgrid[house_low_x+9][y][house_high_z-1][0]=inout_wallcorner_tile_l;
worldgrid[house_low_x+9][y][house_high_z-1][1]=0;
worldgrid[house_low_x+10][y][house_high_z-1][0]=inout_wallcorner_tile_r;
worldgrid[house_low_x+10][y][house_high_z-1][1]=0;
worldgrid[house_low_x+11][y][house_high_z-1][0]=window_tile;
worldgrid[house_low_x+11][y][house_high_z-1][1]=1;
worldgrid[house_low_x+12][y][house_high_z-1][0]=inout_wallcorner_tile_l;
```

```
 worldgrid[house_low_x+12][y][house_high_z-1][1]=0;
worldgrid[house_low_x+13][y][house_high_z-1][0]=wall_tile;
worldgrid[house_low_x+13][y][house_high_z-1][1]=1;
//3 inside
worldgrid[house_low_x+1][y][house_high_z-2][0]=wall_tile;
worldgrid[house_low_x+1][y][house_high_z-2][1]=0;
worldgrid[house_low_x+2][y][house_high_z-2][0]=block_tile;
worldgrid[house_low_x+2][y][house_high_z-2][1]=2;
worldgrid[house_low_x+3][y][house_high_z-2][0]=interior_wallcorner_tile_r;
worldgrid[house_low_x+3][y][house_high_z-2][1]=2;
worldgrid[house_low_x+4][y][house_high_z-2][0]=door_tile;
worldgrid[house_low_x+4][y][house_high_z-2][1]=3;
worldgrid[house_low_x+5][y][house_high_z-2][0]=interior_wallcorner_tile_l;
worldgrid[house_low_x+5][y][house_high_z-2][1]=2;
worldgrid[house_low_x+6][y][house_high_z-2][0]=interior_wallcorner_tile_r;
worldgrid[house_low_x+6][y][house_high_z-2][1]=2;
worldgrid[house_low_x+9][y][house_high_z-2][0]=interior_wallcorner_tile_l;
worldgrid[house_low_x+9][y][house_high_z-2][1]=2;
worldgrid[house_low_x+10][y][house_high_z-2][0]=interior_wallcorner_tile_r;
worldgrid[house_low_x+10][y][house_high_z-2][1]=2;
worldgrid[house_low_x+11][y][house_high_z-2][0]=door_tile;
worldgrid[house_low_x+11][y][house_high_z-2][1]=3;
worldgrid[house_low_x+12][y][house_high_z-2][0]=interior_wallcorner_tile_l;
worldgrid[house_low_x+12][y][house_high_z-2][1]=2;

//4 outside
worldgrid[house_high_x-1][y][house_high_z-1][0]=wallcorner_tile;
worldgrid[house_high_x-1][y][house_high_z-1][1]=1;
worldgrid[house_high_x-1][y][house_high_z-2][0]=wall_tile;
worldgrid[house_high_x-1][y][house_high_z-2][1]=2;
worldgrid[house_high_x-1][y][house_high_z-3][0]=inout_wallcorner_tile_r;
worldgrid[house_high_x-1][y][house_high_z-3][1]=1;
worldgrid[house_high_x-1][y][house_high_z-4][0]=window_tile;
worldgrid[house_high_x-1][y][house_high_z-4][1]=2;
worldgrid[house_high_x-1][y][house_high_z-5][0]=inout_wallcorner_tile_l;
worldgrid[house_high_x-1][y][house_high_z-5][1]=1;
worldgrid[house_high_x-1][y][house_high_z-6][0]=inout_wallcorner_tile_r;
worldgrid[house_high_x-1][y][house_high_z-6][1]=1;
worldgrid[house_high_x-1][y][house_high_z-8][0]=double_window_tile;
worldgrid[house_high_x-1][y][house_high_z-8][1]=0;
worldgrid[house_high_x-1][y][house_high_z-9][0]=inout_wallcorner_tile_l;
worldgrid[house_high_x-1][y][house_high_z-9][1]=1;
worldgrid[house_high_x-1][y][house_high_z-10][0]=inout_wallcorner_tile_r;
worldgrid[house_high_x-1][y][house_high_z-10][1]=1;
worldgrid[house_high_x-1][y][house_high_z-11][0]=window_tile;
worldgrid[house_high_x-1][y][house_high_z-11][1]=2;
worldgrid[house_high_x-1][y][house_high_z-12][0]=inout_wallcorner_tile_l;
worldgrid[house_high_x-1][y][house_high_z-12][1]=1;
worldgrid[house_high_x-1][y][house_high_z-14][0]=wallcorner_tile;
worldgrid[house_high_x-1][y][house_high_z-14][1]=2;
worldgrid[house_high_x-1][y][house_high_z-13][0]=wall_tile;
```

```
    worldgrid[house_high_x-1][y][house_high_z-13][1]=2;
    //4 inside
    worldgrid[house_high_x-2][y][house_high_z-1][0]=wall_tile;
    worldgrid[house_high_x-2][y][house_high_z-1][1]=1;
    worldgrid[house_high_x-2][y][house_high_z-2][0]=block_tile;
    worldgrid[house_high_x-2][y][house_high_z-2][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-3][0]=interior_wallcorner_tile_r;
    worldgrid[house_high_x-2][y][house_high_z-3][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-4][0]=door_tile;
    worldgrid[house_high_x-2][y][house_high_z-4][1]=0;
    worldgrid[house_high_x-2][y][house_high_z-5][0]=interior_wallcorner_tile_l;
    worldgrid[house_high_x-2][y][house_high_z-5][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-6][0]=interior_wallcorner_tile_r;
    worldgrid[house_high_x-2][y][house_high_z-6][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-9][0]=interior_wallcorner_tile_l;
    worldgrid[house_high_x-2][y][house_high_z-9][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-10][0]=interior_wallcorner_tile_r;
    worldgrid[house_high_x-2][y][house_high_z-10][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-11][0]=door_tile;
    worldgrid[house_high_x-2][y][house_high_z-11][1]=0;
    worldgrid[house_high_x-2][y][house_high_z-12][0]=interior_wallcorner_tile_l;
    worldgrid[house_high_x-2][y][house_high_z-12][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-14][0]=wall_tile;
    worldgrid[house_high_x-2][y][house_high_z-14][1]=3;
    worldgrid[house_high_x-2][y][house_high_z-13][0]=block_tile;
    worldgrid[house_high_x-2][y][house_high_z-13][1]=1;
}

if(allocentric){
    //G
    a_win_pos[0]=clue_win_array[4][1];
    a_win_pos[1]=clue_win_array[4][2];
    a_win_pos[2]=clue_win_array[4][3];
    //G

    //D
    b_win_pos[0]=clue_win_array[2][4];
    b_win_pos[1]=clue_win_array[2][5];
    b_win_pos[2]=clue_win_array[2][6];
    //D

    //P
    c_win_pos[0]=clue_win_array[7][1];
    c_win_pos[1]=clue_win_array[7][2];
    c_win_pos[2]=clue_win_array[7][3];
    //P

worldgrid[a_win_pos[0]][a_win_pos[1]][a_win_pos[2]][0]=circle_window_tile;
            worldgrid[b_win_pos[0]][b_win_pos[1]][b_win_pos[2]][0]=X_window_tile;
            worldgrid[c_win_pos[0]][c_win_pos[1]][c_win_pos[2]][0]=triangle_window_tile;
```

```
    switch(targetwin+1){
            case 1:
                    //climb the stairs
                    expectdist = clue_win_array[6][7];
                    //for the clue chart resembles letter W
                    win_clue=clue_win_array[6][0];
            break;
            case 2:
                    //climb the stairs
                    expectdist = clue_win_array[2][8];
                    //for the clue chart resembles letter F
                    win_clue=clue_win_array[2][0]+2;
            break;
            case 3:
                    //climb the stairs
                    expectdist = clue_win_array[7][7];
                    //for the clue chart resembles letter H
                    win_clue=clue_win_array[7][0];
            break;
    }
}
else{

    switch(trial){
            case 1:

                    //climb the stairs
                    //1
                    if((int)rand()%2){
                            a_win_pos[0]=clue_win_array[1][1];
                            a_win_pos[1]=clue_win_array[1][2];
                            a_win_pos[2]=clue_win_array[1][3];
                            win_clue=clue_win_array[1][0];
                            expectdist = clue_win_array[1][7];
                    }
                    else{
                            a_win_pos[0]=clue_win_array[1][4];
                            a_win_pos[1]=clue_win_array[1][5];
                            a_win_pos[2]=clue_win_array[1][6];
                            win_clue=clue_win_array[1][0]+2;
                            expectdist = clue_win_array[1][8];
                    }
                    //1

                    //to the target window
                    //4
                    if((int)rand()%2){
                            b_win_pos[0]=clue_win_array[4][1];
                            b_win_pos[1]=clue_win_array[4][2];
                            b_win_pos[2]=clue_win_array[4][3];
                    }
```

```
		else{
			b_win_pos[0]=clue_win_array[4][4];
			b_win_pos[1]=clue_win_array[4][5];
			b_win_pos[2]=clue_win_array[4][6];
		}
		//4

		//3
		if((int)rand()%2){
			c_win_pos[0]=clue_win_array[3][1];
			c_win_pos[1]=clue_win_array[3][2];
			c_win_pos[2]=clue_win_array[3][3];
		}
		else{
			c_win_pos[0]=clue_win_array[3][4];
			c_win_pos[1]=clue_win_array[3][5];
			c_win_pos[2]=clue_win_array[3][6];
		}
		//3

break;
case 2:
		//climb the stairs
		//2
		if((int)rand()%2){
			a_win_pos[0]=clue_win_array[2][1];
			a_win_pos[1]=clue_win_array[2][2];
			a_win_pos[2]=clue_win_array[2][3];
			win_clue=clue_win_array[2][0];
			expectdist = clue_win_array[2][7];
		}
		else{
			a_win_pos[0]=clue_win_array[2][4];
			a_win_pos[1]=clue_win_array[2][5];
			a_win_pos[2]=clue_win_array[2][6];
			win_clue=clue_win_array[2][0]+2;
			expectdist = clue_win_array[2][8];
		}
		//2

		//to the target window
		//7
		if((int)rand()%2){
			b_win_pos[0]=clue_win_array[7][1];
			b_win_pos[1]=clue_win_array[7][2];
			b_win_pos[2]=clue_win_array[7][3];
		}
		else{
			b_win_pos[0]=clue_win_array[7][4];
			b_win_pos[1]=clue_win_array[7][5];
			b_win_pos[2]=clue_win_array[7][6];
```

```
		}
		//7

		//8
		if((int)rand()%2){
			c_win_pos[0]=clue_win_array[8][1];
			c_win_pos[1]=clue_win_array[8][2];
			c_win_pos[2]=clue_win_array[8][3];
		}
		else{
			c_win_pos[0]=clue_win_array[8][4];
			c_win_pos[1]=clue_win_array[8][5];
			c_win_pos[2]=clue_win_array[8][6];
		}
		//8
break;
case 3:
		//climb the stairs
		//3
		if((int)rand()%2){
			a_win_pos[0]=clue_win_array[3][1];
			a_win_pos[1]=clue_win_array[3][2];
			a_win_pos[2]=clue_win_array[3][3];
			win_clue=clue_win_array[3][0];
			expectdist = clue_win_array[3][7];
		}
		else{
			a_win_pos[0]=clue_win_array[3][4];
			a_win_pos[1]=clue_win_array[3][5];
			a_win_pos[2]=clue_win_array[3][6];
			win_clue=clue_win_array[3][0]+2;
			expectdist = clue_win_array[3][8];
		}
		//3

		//to the target window
		//6
		if((int)rand()%2){
			b_win_pos[0]=clue_win_array[6][1];
			b_win_pos[1]=clue_win_array[6][2];
			b_win_pos[2]=clue_win_array[6][3];
		}
		else{
			b_win_pos[0]=clue_win_array[6][4];
			b_win_pos[1]=clue_win_array[6][5];
			b_win_pos[2]=clue_win_array[6][6];
		}
		//6

		//5
		if((int)rand()%2){
```

```
                c_win_pos[0]=clue_win_array[5][1];
                c_win_pos[1]=clue_win_array[5][2];
                c_win_pos[2]=clue_win_array[5][3];
        }
        else{
                c_win_pos[0]=clue_win_array[5][4];
                c_win_pos[1]=clue_win_array[5][5];
                c_win_pos[2]=clue_win_array[5][6];
        }
        //5

break;
case 4:
        //climb the stairs
        //4
        if((int)rand()%2){
                a_win_pos[0]=clue_win_array[4][1];
                a_win_pos[1]=clue_win_array[4][2];
                a_win_pos[2]=clue_win_array[4][3];
                win_clue=clue_win_array[4][0];
                expectdist = clue_win_array[4][7];
        }
        else{
                a_win_pos[0]=clue_win_array[4][4];
                a_win_pos[1]=clue_win_array[4][5];
                a_win_pos[2]=clue_win_array[4][6];
                win_clue=clue_win_array[4][0]+2;
                expectdist = clue_win_array[4][8];
        }
        //4

        //to the target window
        //3
        if((int)rand()%2){
                b_win_pos[0]=clue_win_array[3][1];
                b_win_pos[1]=clue_win_array[3][2];
                b_win_pos[2]=clue_win_array[3][3];
        }
        else{
                b_win_pos[0]=clue_win_array[3][4];
                b_win_pos[1]=clue_win_array[3][5];
                b_win_pos[2]=clue_win_array[3][6];
        }
        //3

        //6
        if((int)rand()%2){
                c_win_pos[0]=clue_win_array[6][1];
                c_win_pos[1]=clue_win_array[6][2];
                c_win_pos[2]=clue_win_array[6][3];
        }
```

```
		else{
			c_win_pos[0]=clue_win_array[6][4];
			c_win_pos[1]=clue_win_array[6][5];
			c_win_pos[2]=clue_win_array[6][6];
		}
		//6

break;
case 5:
		//climb the stairs
		//5
		if((int)rand()%2){
			a_win_pos[0]=clue_win_array[5][1];
			a_win_pos[1]=clue_win_array[5][2];
			a_win_pos[2]=clue_win_array[5][3];
			win_clue=clue_win_array[5][0];
			expectdist = clue_win_array[5][7];
		}
		else{
			a_win_pos[0]=clue_win_array[5][4];
			a_win_pos[1]=clue_win_array[5][5];
			a_win_pos[2]=clue_win_array[5][6];
			win_clue=clue_win_array[5][0]+2;
			expectdist = clue_win_array[5][8];
		}
		//5

		//to the target window
		//1
		if((int)rand()%2){
			b_win_pos[0]=clue_win_array[1][1];
			b_win_pos[1]=clue_win_array[1][2];
			b_win_pos[2]=clue_win_array[1][3];
		}
		else{
			b_win_pos[0]=clue_win_array[1][4];
			b_win_pos[1]=clue_win_array[1][5];
			b_win_pos[2]=clue_win_array[1][6];
		}
		//1

		//4
		if((int)rand()%2){
			c_win_pos[0]=clue_win_array[4][1];
			c_win_pos[1]=clue_win_array[4][2];
			c_win_pos[2]=clue_win_array[4][3];
		}
		else{
			c_win_pos[0]=clue_win_array[4][4];
			c_win_pos[1]=clue_win_array[4][5];
			c_win_pos[2]=clue_win_array[4][6];
```

```
		}
		//4

break;
case 6:
		//climb the stairs
		//6
		if((int)rand()%2){
			a_win_pos[0]=clue_win_array[6][1];
			a_win_pos[1]=clue_win_array[6][2];
			a_win_pos[2]=clue_win_array[6][3];
			win_clue=clue_win_array[6][0];
			expectdist = clue_win_array[6][7];
		}
		else{
			a_win_pos[0]=clue_win_array[6][4];
			a_win_pos[1]=clue_win_array[6][5];
			a_win_pos[2]=clue_win_array[6][6];
			win_clue=clue_win_array[6][0]+2;
			expectdist = clue_win_array[6][8];
		}
		//6

		//to the target window
		//8
		if((int)rand()%2){
			b_win_pos[0]=clue_win_array[8][1];
			b_win_pos[1]=clue_win_array[8][2];
			b_win_pos[2]=clue_win_array[8][3];
		}
		else{
			b_win_pos[0]=clue_win_array[8][4];
			b_win_pos[1]=clue_win_array[8][5];
			b_win_pos[2]=clue_win_array[8][6];
		}
		//8

		//2
		if((int)rand()%2){
			c_win_pos[0]=clue_win_array[2][1];
			c_win_pos[1]=clue_win_array[2][2];
			c_win_pos[2]=clue_win_array[2][3];
		}
		else{
			c_win_pos[0]=clue_win_array[2][4];
			c_win_pos[1]=clue_win_array[2][5];
			c_win_pos[2]=clue_win_array[2][6];
		}
		//2

break;
```

```
case 7:
	//climb the stairs
	//7
	if((int)rand()%2){
		a_win_pos[0]=clue_win_array[7][1];
		a_win_pos[1]=clue_win_array[7][2];
		a_win_pos[2]=clue_win_array[7][3];
		win_clue=clue_win_array[7][0];
		expectdist = clue_win_array[7][7];
	}
	else{
		a_win_pos[0]=clue_win_array[7][4];
		a_win_pos[1]=clue_win_array[7][5];
		a_win_pos[2]=clue_win_array[7][6];
		win_clue=clue_win_array[7][0]+2;
		expectdist = clue_win_array[7][8];
	}
	//7

	//to the target window
	//5
	if((int)rand()%2){
		b_win_pos[0]=clue_win_array[5][1];
		b_win_pos[1]=clue_win_array[5][2];
		b_win_pos[2]=clue_win_array[5][3];
	}
	else{
		b_win_pos[0]=clue_win_array[5][4];
		b_win_pos[1]=clue_win_array[5][5];
		b_win_pos[2]=clue_win_array[5][6];
	}
	//5

	//3
	if((int)rand()%2){
		c_win_pos[0]=clue_win_array[3][1];
		c_win_pos[1]=clue_win_array[3][2];
		c_win_pos[2]=clue_win_array[3][3];
	}
	else{
		c_win_pos[0]=clue_win_array[3][4];
		c_win_pos[1]=clue_win_array[3][5];
		c_win_pos[2]=clue_win_array[3][6];
	}
	//3

break;
case 8:
	//climb the stairs
	//8
	if((int)rand()%2){
```

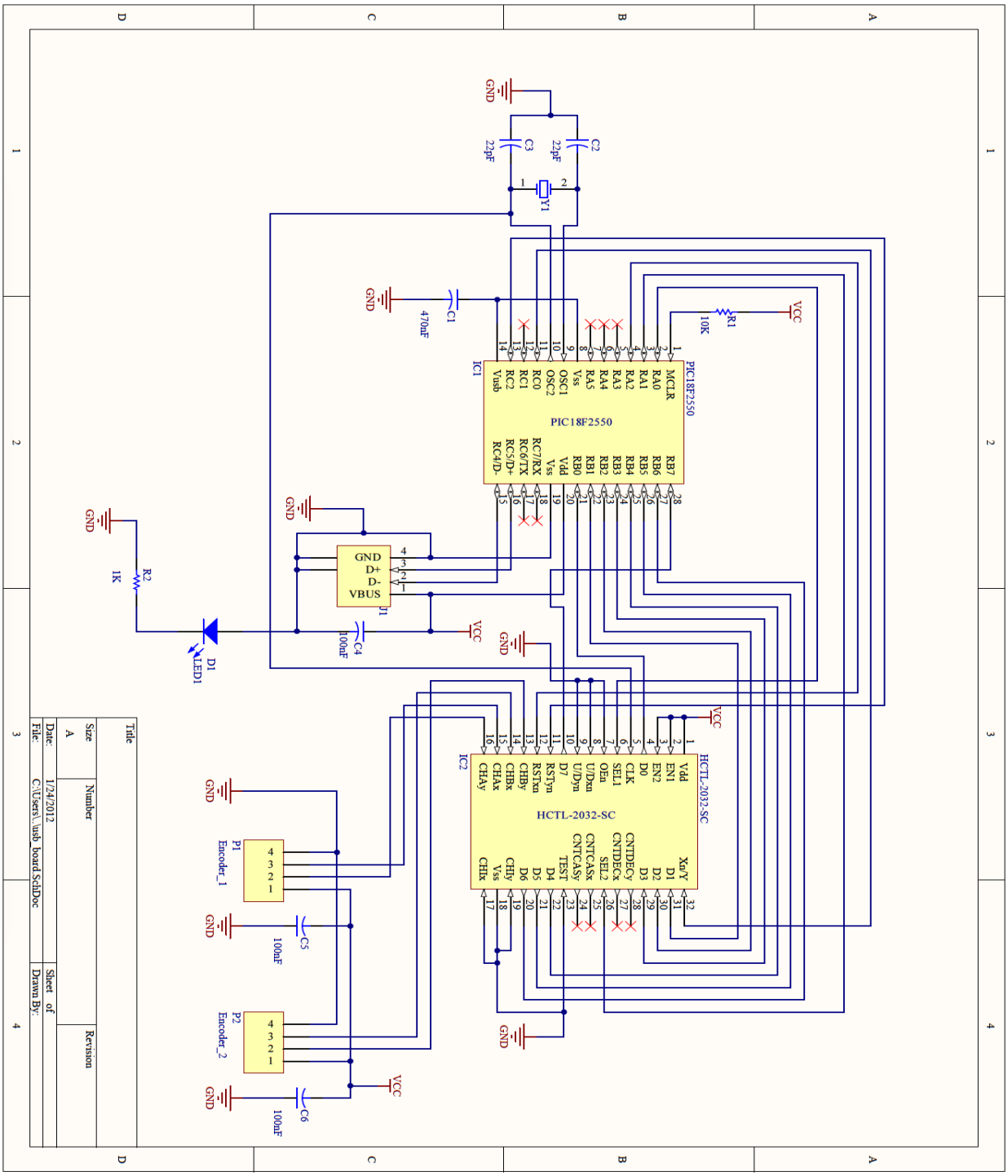
```
                        a_win_pos[0]=clue_win_array[8][1];
                        a_win_pos[1]=clue_win_array[8][2];
                        a_win_pos[2]=clue_win_array[8][3];
                        win_clue=clue_win_array[8][0];
                        expectdist = clue_win_array[8][7];
                }
                else{
                        a_win_pos[0]=clue_win_array[8][4];
                        a_win_pos[1]=clue_win_array[8][5];
                        a_win_pos[2]=clue_win_array[8][6];
                        win_clue=clue_win_array[8][0]+2;
                        expectdist = clue_win_array[8][8];
                }
                //8

                //to the target window
                //2
                if((int)rand()%2){
                        b_win_pos[0]=clue_win_array[2][1];
                        b_win_pos[1]=clue_win_array[2][2];
                        b_win_pos[2]=clue_win_array[2][3];
                }
                else{
                        b_win_pos[0]=clue_win_array[2][4];
                        b_win_pos[1]=clue_win_array[2][5];
                        b_win_pos[2]=clue_win_array[2][6];
                }
                //2

                //6
                if((int)rand()%2){
                        c_win_pos[0]=clue_win_array[6][1];
                        c_win_pos[1]=clue_win_array[6][2];
                        c_win_pos[2]=clue_win_array[6][3];
                }
                else{
                        c_win_pos[0]=clue_win_array[6][4];
                        c_win_pos[1]=clue_win_array[6][5];
                        c_win_pos[2]=clue_win_array[6][6];
                }
                //6

        break;
}

        switch(targetwin)   //targetwin
        {
        case 0:
worldgrid[a_win_pos[0]][a_win_pos[1]][a_win_pos[2]][0]=circle_window_tile;
                worldgrid[b_win_pos[0]][b_win_pos[1]][b_win_pos[2]][0]=X_window_tile;
```

```
        worldgrid[c_win_pos[0]][c_win_pos[1]][c_win_pos[2]][0]=triangle_window_tile;
                                        break;

                case 1:worldgrid[a_win_pos[0]][a_win_pos[1]][a_win_pos[2]][0]=X_window_tile;

        worldgrid[b_win_pos[0]][b_win_pos[1]][b_win_pos[2]][0]=circle_window_tile;

        worldgrid[c_win_pos[0]][c_win_pos[1]][c_win_pos[2]][0]=triangle_window_tile;
                                        break;

                case 2:
        worldgrid[a_win_pos[0]][a_win_pos[1]][a_win_pos[2]][0]=triangle_window_tile;
                                worldgrid[b_win_pos[0]][b_win_pos[1]][b_win_pos[2]][0]=X_window_tile;

        worldgrid[c_win_pos[0]][c_win_pos[1]][c_win_pos[2]][0]=circle_window_tile;
                                        break;
                }

        }
}
```

End of “genworldgrid.h” ----------

Appendix C

Schematic design of the VRNChair motion capture.



Appendix D

VRNChair C++ code, written for the Microchip PIC18F2550 microcontroller.

Start of “main.c” ----------

```
#include <18F2550.h>
#include <math.h>

//~~~ 20MHZ OSCILLATOR CONFIGS ~~~//
// FULL SPEED
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV1,VREGEN
#use delay(clock=48000000)
#define USB_USE_FULL_SPEED  TRUE

/*
////SLOW SPEED
#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,NODEBUG,USBDIV,PLL5,CPUDIV3,VREGEN
#use delay(clock=24000000)
#define USB_USE_FULL_SPEED  FALSE
*/

#DEFINE USB_HID_DEVICE  TRUE  //Tells the CCS PIC USB firmware to include HID handling code.

//turn on EP1 for IN interrupt transfers.  (IN = PIC -> PC)
#define USB_EP1_TX_ENABLE  USB_ENABLE_INTERRUPT
#define USB_EP1_TX_SIZE 8     //max packet size of this endpoint

//turn on EP2 for OUT interrupt transfers.  (OUT = PC -> PIC)
#define USB_EP2_RX_ENABLE  USB_ENABLE_INTERRUPT
#define USB_EP2_RX_SIZE 8     //max packet size of this endpoint

#include <lpic18_usb.h>      //Microchip PIC18Fxx5x hardware layer for usb.c
#include "main.h"    //USB Configuration and Device descriptors for this UBS device
#include <lusb.c>           //handles usb setup tokens and get descriptor reports
#include <string.h>

//Pin definitions
#define enc_data     input_b
#define enc_sel1     PIN_A0
#define enc_sel2     PIN_A1
#define enc_1_rst    PIN_A2
#define enc_1_oe     PIN_C0
#define enc_2_rst    PIN_C2
#define enc_2_oe     PIN_C1
```

```
#define upN_down     PIN_C7
#define leftN_right  PIN_C6

//int8 arr_usb_data_out_raw[8];
int8 arr_usb_data_out[8];

float time,x,y;

//filter memory allocation
#define filter_size  4
unsigned int8 data_joy[filter_size][2];
unsigned int8 write_point=0, filter_count;
unsigned int16 filter_temp_x, filter_temp_y;
unsigned int8 out_x,out_y;

// fires when the timer rolls over
#INT_TIMER3
void timer3_isr() {

}


// external interrupt when pulsetrain goes high
#INT_EXT
void ext_isr() {
}

void write_eeprom_settings() {
}

void get_eeprom_settings() {
}

void init_enc(){
  set_tris_b(0xff);
  output_low(enc_1_rst);
  output_low(enc_2_rst);
  output_low(enc_sel1);
  output_low(enc_sel2);
  output_high(enc_1_rst);
  output_high(enc_2_rst);
  output_high(enc_1_oe);
  output_high(enc_2_oe);
}

signed int32 read_enc(int axis){
  int32 value;
  int8 byte1,byte2,byte3,byte4;
```

```
output_high(enc_1_oe);
output_high(enc_2_oe);

delay_ms(10);   //according to the datasheet

output_low(enc_sel1); output_high(enc_sel2);

if(axis==1) output_low(enc_1_oe);
if(axis==2) output_low(enc_2_oe);

// check stability of the data for byte 1 (highest)
do{
  byte1=enc_data();
}while(byte1!=enc_data());

output_high(enc_sel1); output_high(enc_sel2);

// check stability of the data for byte 2
do{
  byte2=enc_data();
}while(byte2!=enc_data());

output_low(enc_sel1); output_low(enc_sel2);

// check stability of the data for byte 3
do{
  byte3=enc_data();
}while(byte3!=enc_data());

output_high(enc_sel1); output_low(enc_sel2);

// check stability of the data for byte 4 (lowest)
do{
  byte4=enc_data();
}while(byte4!=enc_data());

output_high(enc_1_oe);
output_high(enc_2_oe);

delay_us(25);   //according to the datasheet

value=(int32)byte1;
value<<=8;
value|=(int32)(byte2);
value<<=8;
value|=(int32)(byte3);
value<<=8;
```

```
  value|=(int32)(byte4);

  //calculation for negative value
  if((value>>31))
    value=-(~value);
  return value;
}

void reset_enc(int axis){
  if(axis==1){
    output_low(enc_1_rst);
    delay_us(1);
    output_high(enc_1_rst);
  }
  if(axis==2){
    output_low(enc_2_rst);
    delay_us(1);
    output_high(enc_2_rst);
  }
}



////////////////////////////////////////////////////////////////////////////
// Main application code
////////////////////////////////////////////////////////////////////////////
void main(void) {

 Byte Data_in,Data_out;
 int8 Buttons = 0;

 signed int16 vel_l,vel_r;
 signed int16 comp_x,comp_y;

 setup_adc_ports(NO_ANALOGS);
 setup_adc(ADC_OFF);
 setup_spi(FALSE);

 output_float(upN_down);
 output_float(leftN_right);

 init_enc();

 // this timer will overflow in 22ms
 // 65536 pulses per 22ms
 // therefore each count is 22000us / 65536 =0.3 us
 // lets test this 1 ms = 1000/0.3 = 3333 pulses thus we need to preload with 65536 - 3333 = 62203 to
get a 1ms pulse
```

```
//   set_timer3(65260); // = 100us   = 276 pulses
//   set_timer3(65200); // = 120us   = 336 pulses
//   set_timer3(65110); // = 150us   = 426 pulses
//   set_timer3(65000); // = 180us   = 536 pulses
//   set_timer3(62580); // = 1 ms    = 2950 pulses
//   set_timer3(62203); // = 1.12 ms = 3333 pulses


  // which is
  // setup_timer_3(T3_INTERNAL|T3_DIV_BY_2);   // PPM long timer


//  ext_int_edge(L_TO_H);      // init interrupt triggering for button press
//  enable_interrupts(INT_EXT);// turn on external interrupts

  enable_interrupts(GLOBAL);

  usb_init_cs();


   while (TRUE) {
     usb_task();

     if (usb_enumerated()) {

       if(time<6283)
         time+=1;
       else
         time=0;

       vel_r=read_enc(2);
       reset_enc(1);

       vel_l=read_enc(1);
       reset_enc(2);

       comp_y=-1*(signed int16)(vel_l+vel_r)/4;
       comp_x=(signed int16)(vel_l-vel_r)/2;

       output_float(upN_down);
       if(comp_y>20)  output_low(upN_down);
       if(comp_y<-20) output_high(upN_down);

       output_float(leftN_right);
       if(comp_x>40)  output_low(leftN_right);
       if(comp_x<-40) output_high(leftN_right);

       if(comp_y>252)  comp_y=252;
```

```
      if(comp_y<-252) comp_y=-252;
      if(comp_x>252)  comp_x=252;
      if(comp_x<-252) comp_x=-252;

      // The corrected joystick data is output on Report Id 1
      arr_usb_data_out[0] = 0x01;

   //    x=(int8)(255*(1+sin(time/500))/2);
   //    y=(int8)(255*(1+cos(time/500))/2);

      //low pass filtering
      if(write_point<(filter_size-1))
        write_point++;
      else  write_point = 0;

      data_joy[write_point][0]=(unsigned int8)(128-(comp_x/2));
      data_joy[write_point][1]=(unsigned int8)(128-(comp_y/2));
      filter_temp_x=0;
      filter_temp_y=0;
      for(filter_count=0;filter_count<filter_size;filter_count++){
        filter_temp_x+=(unsigned int16)data_joy[filter_count][0];
        filter_temp_y+=(unsigned int16)data_joy[filter_count][1];
      }
      out_x=(unsigned int8)(filter_temp_x>>2);
      out_y=(unsigned int8)(filter_temp_y>>2);
   //    if((out_x<148)&&(out_x>108))
   //       out_x=128;
      //end of low pass filtering
      arr_usb_data_out[1] = out_x;
      arr_usb_data_out[2] = out_y;
      /*
      arr_usb_data_out[3] = 128;
      arr_usb_data_out[4] = y;
      arr_usb_data_out[5] = x;
      arr_usb_data_out[6] = y;
      arr_usb_data_out[7] = (int8)x/16;
      */
      arr_usb_data_out[3] = 0;
      arr_usb_data_out[4] = 0;
      arr_usb_data_out[5] = 0;
      arr_usb_data_out[6] = 0;
      arr_usb_data_out[7] = 0;
      usb_put_packet(1,arr_usb_data_out,8,USB_DTS_TOGGLE);
    }
  }
}
```

End of “main.c” ----------

Start of “main.h” ----------

```
#IFNDEF __USB_DESCRIPTORS__
#DEFINE __USB_DESCRIPTORS__

#include <usb.h>

  ////////////////////////////////////////////////////////////////
  ///
  /// HID Report.  Tells HID driver how to handle and deal with
  /// received data.  HID Reports can be extremely complex,
  /// see HID specifcation for help on writing your own.
  ///
  /// This examples configures HID driver to take received data
  /// as mouse x, y and button data.
  ///
  ////////////////////////////////////////////////////////////////

  const char USB_CLASS_SPECIFIC_DESC[] = {
   // Report 1
   0x05, 0x01,      // Usage Page (Generic Desktop)         1,  2
   0x09, 0x04,      // Usage (Joystick)                     3,  4
   0xA1, 0x01,      // Collection (Application)             5,  6
   0x85, 0x01,      // Report Id (1)                        7,  8
   0x09, 0x01,      // Usage (Pointer)                      9, 10
   0xA1, 0x00,      // Collection (Physical)               11, 12
   0x09, 0x30,      // Usage(X)                            13, 14
   0x09, 0x31,      // Usage(Y)                            15, 16
   0x09, 0x32,      // Usage(Z)                            17, 18
   0x09, 0x33,      // Usage(Rx)                           19, 20
   0x09, 0x34,      // Usage(Ry)                           21, 22
   0x09, 0x35,      // Usage(Rz)                           23, 24
   0x15, 0x00,      // Logical Minimum (0)                 25, 26
   0x26, 0xFF, 0x00, // Logical Maximum (FF)               27, 28, 29
   0x35, 0x00,      // Physical Minimum (0)                30, 31
   0x46, 0xFF, 0x00, // Physical Maximum (FF)              32, 33, 34
   0x95, 0x06,      // Report Count (6)                    35, 36
   0x75, 0x08,      // Report Size (8)                     37, 38
   0x81, 0x02,      // Input (Data, Variable, Absolute)    39, 40
   0xC0,          // End Collection                        41
   0x05, 0x09,      // Usage Page (Button)                 42, 43
   0x19, 0x01,      // Usage Minimum (01)                  44, 45
```

```
0x29, 0x05,      // Usage Maximum (05)                46, 47
0x15, 0x00,      // Logical Minimum (0)               48, 49
0x25, 0x01,      // Logical Maximum (Toggle)          50, 51
0x35, 0x00,      // Physical Minimum (0)              52, 53
0x45, 0x01,      // Physical Maximum (0)              54, 55
0x75, 0x01,      // Report Size (1)                   56, 57
0x95, 0x04,      // Report Count (4)                  58, 59
0x81, 0x02,      // Input (Data, Variable, Absolute)  60, 61
0x75, 0x01,      // Report Size (1)                   62, 63
0x95, 0x04,      // Report Count (4)                  64, 65
0x81, 0x01,      // Input (Data, Variable, Absolute)  66, 67
0xC0,          // End Collection                    68
// Report 2
0x05, 0x01,      // Usage Page (Generic Desktop)      69, 70
0x09, 0x08,      // Usage (Multi-axis Controller)     71, 72
0xA1, 0x01,      // Collection (Application)          73, 74
0x85, 0x02,      // Report Id (2)                     75, 76
0x09, 0x01,      // Usage (Pointer)                   77, 78
0xA1, 0x00,      // Collection (Physical)             79, 80
0x09, 0x30,      // Usage(X)                          81, 82
0x09, 0x31,      // Usage(Y)                          83, 84
0x09, 0x32,      // Usage(Z)                          85, 86
0x09, 0x33,      // Usage(Rx)                         87, 88
0x09, 0x34,      // Usage(Ry)                         89, 90
0x09, 0x35,      // Usage(Rz)                         91, 92
0x09, 0x36,      // Usage(Slider)                     93, 94
0x15, 0x00,      // Logical Minimum (0)               95, 96
0x26, 0xFF, 0x00, // Logical Maximum (FF)             97, 98, 99
0x35, 0x00,      // Physical Minimum (0)              100,101
0x46, 0xFF, 0x00, // Physical Maximum (0)             102,103,104
0x95, 0x07,      // Report Count (7)                  105,106
0x75, 0x08,      // Report Size (8)                   107,108
0x81, 0x02,      // Input (Data, Variable, Absolute)  109,110
0xC0,          // End Collection                    111
0xC0,          // End Collection                    112
// Report 3
0x05, 0x01,      // Usage Page (Generic Desktop)      113,114
0x09, 0x08,      // Usage (Multi-axis Controller)     115,116
0xA1, 0x01,      // Collection (Application)          117,118
0x85, 0x03,      // Report Id (3)                     119,120
0x09, 0x01,      // Usage (Pointer)                   121,122
0xA1, 0x00,      // Collection (Physical)             123,124
```

```
0x09, 0x30,      // Usage(X)                    125,126
0x09, 0x31,      // Usage(Y)                    127,128
0x09, 0x32,      // Usage(Z)                    129,130
0x09, 0x33,      // Usage(Rx)                   131,132
0x09, 0x34,      // Usage(Ry)                   133,134
0x09, 0x35,      // Usage(Rz)                   135,136
0x09, 0x36,      // Usage(Slider)               137,138
0x15, 0x00,      // Logical Minimum (0)         139,140
0x26, 0xFF, 0x00, // Logical Maximum (FF)       141,142,143
0x35, 0x00,      // Physical Minimum (0)        144,145
0x46, 0xFF, 0x00, // Physical Maximum (0)       146,147,148
0x95, 0x07,      // Report Count (7)            149,150
0x75, 0x08,      // Report Size (8)             151,152
0x81, 0x02,      // Input (Data, Variable, Absolute)  153,154
0xC0,            // End Collection              155
0x05, 0x0F,      // Usage Page (PID)            156,157
0xA1, 0x02,      // Collection (Logical)        158,159
0x09, 0x55,      // Usage (Reserved)            160,161
0x05, 0x01,      // Usage Page (Generic Desktop)      162,163
0x09, 0x01,      // Usage (Pointer)             164,165
0xA1, 0x00,      // Collection (Physical)       166,167
0x09, 0x30,      // Usage(Usage_X)              168,169
0x09, 0x31,      // Usage(Usage_Y)              170,171
0x09, 0x32,      // Usage(Z)                    172,173
0x09, 0x33,      // Usage(Rx)                   174,175
0x09, 0x34,      // Usage(Ry)                   176,177
0x09, 0x35,      // Usage(Rz)                   178,179
0x09, 0x6D,      // Usage(Reserved)             180,181
0x26, 0xFF, 0x00, // Logical Maximum (FF)       182,183,184
0x75, 0x08,      // Report Size (8)             185,186
0x95, 0x07,      // Report Count (7)            187,188
0x91, 0x02,      // Output (Data, Variable, Absolute)  189,190
0xC0,            // End Collection              191
0xC0,            // End Collection              192
0xC0,            // End Collection              193
// Report 4
0x05, 0x01,      // Usage Page (Generic Desktop)      194,195
0x09, 0x08,      // Usage (Multi-axis Controller)     196,197
0xA1, 0x01,      // Collection (Application)    198,199
0x85, 0x04,      // Usage Page                  200,201
0x09, 0x01,      // Usage (Pointer)             202,203
0xA1, 0x00,      // Collection (Physical)       204,205
```

```
0x09, 0x30,      // Usage(X)                    206,207
0x09, 0x31,      // Usage(Y)                    208,209
0x09, 0x32,      // Usage(Z)                    210,211
0x09, 0x33,      // Usage(Rx)                    212,213
0x09, 0x34,      // Usage(Ry)                    214,215
0x09, 0x35,      // Usage(Rz)                    216,217
0x09, 0x36,      // Usage(Slider)                218,219
0x15, 0x00,      // Logical Minimum (0)              220,221
0x26, 0xFF, 0x00, // Logical Maximum (FF)          222,223,224
0x35, 0x00,      // Physical Minimum (0)             225,226
0x46, 0xFF, 0x00, // Physical Maximum (0)          227,228,229
0x95, 0x07,      // Report Count (7)                230,231
0x75, 0x08,      // Report Size (8)                 232,233
0x81, 0x02,      // Input (Data, Variable, Absolute)    234,235
0xC0,           // End Collection                   236
0x05, 0x0F,      // Usage Page (Game Pad)               237,238
0xA1, 0x02,      // Collection (Logical)             239,240
0x09, 0x55,      // Usage (Reserved)                 241,242
0x05, 0x01,      // Usage Page (Generic Desktop)        243,244
0x09, 0x01,      // Usage (Pointer)                 245,246
0xA1, 0x00,      // Collection (Physical)            247,248
0x09, 0x30,      // Usage(Usage_X)                   249,250
0x09, 0x31,      // Usage(Usage_Y)                   251,252
0x09, 0x32,      // Usage(Z)                    253,254
0x09, 0x33,      // Usage(Rx)                    255,256
0x09, 0x34,      // Usage(Ry)                    257,258
0x09, 0x35,      // Usage(Rz)                    259,260
0x09, 0x6D,      // Usage(Slider)                   261,262
0x26, 0xFF, 0x00, // Logical Maximum (FF)          263,264,265
0x75, 0x08,      // Report Size (8)                 266,267
0x95, 0x07,      // Report Count (7)                268,269
0x91, 0x02,      // Input (Data, Variable, Absolute)    270,271
0xC0,           // End Collection                   272
0xC0,           // End Collection                   273
0xC0};          // End Collection                   274  274

//if a class has an extra descriptor not part of the config descriptor,
// this lookup table defines where to look for it in the const
// USB_CLASS_SPECIFIC_DESC[] array.
//first element is the config number (if your device has more than one config)
//second element is which interface number
//set element to 0xFFFF if this config/interface combo doesn't exist
```

```
const int16 USB_CLASS_SPECIFIC_DESC_LOOKUP[USB_NUM_CONFIGURATIONS][1] =
{
//config 1
  //interface 0
    0
};

//if a class has an extra descriptor not part of the config descriptor,
// this lookup table defines the size of that descriptor.
//first element is the config number (if your device has more than one config)
//second element is which interface number
//set element to 0xFFFF if this config/interface combo doesn't exist
const int16 USB_CLASS_SPECIFIC_DESC_LOOKUP_SIZE[USB_NUM_CONFIGURATIONS][1] =
{
//config 1
  //interface 0
  sizeof(USB_CLASS_SPECIFIC_DESC)
};



////////////////////////////////////////////////////////////////
///
///  start config descriptor
///  right now we only support one configuration descriptor.
///  the config, interface, class, and endpoint goes into this array.
///
////////////////////////////////////////////////////////////////

#DEFINE USB_TOTAL_CONFIG_LEN     41 //config+interface+class+endpoint

const char USB_CONFIG_DESC[] = {
//IN ORDER TO COMPLY WITH WINDOWS HOSTS, THE ORDER OF THIS ARRAY MUST BE:
  //   config(s)
  //   interface(s)
  //   class(es)
  //   endpoint(s)

//config_descriptor for config index 1
    USB_DESC_CONFIG_LEN, //length of descriptor size                    ==1
    USB_DESC_CONFIG_TYPE, //constant CONFIGURATION (CONFIGURATION 0x02)      ==2
    USB_TOTAL_CONFIG_LEN,0, //size of all data returned for this config      ==3,4
```

```
    1, //number of interfaces this device supports      ==5
    0x01, //identifier for this configuration.  (IF we had more than one configurations)     ==6
    0x00, //index of string descriptor for this configuration     ==7
    0x80, //bit 6=1 if self powered, bit 5=1 if supports remote wakeup (we don't), bits 0-4 unused and
bit7=1       ==8
    0x10, //maximum bus power required (maximum milliamperes/2)  (0x32 = 100mA)

  //interface descriptor 1
    USB_DESC_INTERFACE_LEN, //length of descriptor     =10
    USB_DESC_INTERFACE_TYPE, //constant INTERFACE (INTERFACE 0x04)      =11
    0x00, //number defining this interface (IF we had more than one interface)    ==12
    0x00, //alternate setting     ==13
    2, //number of endpoins, except 0 (pic167xx has 3, but we dont have to use all).      ==14
    0x03, //class code, 03 = HID     ==15
    0x00, //subclass code //boot     ==16
    0x00, //protocol code      ==17
    0x00, //index of string descriptor for interface      ==18

  //class descriptor 1  (HID)
    USB_DESC_CLASS_LEN, //length of descriptor    ==19
    USB_DESC_CLASS_TYPE, //dscriptor type (0x21 == HID)     ==20
    0x10,0x01, //hid class release number (1.0) (try 1.10)     ==21,22
    0x21, //localized country code (0 = none)      ==23
    0x01, //number of hid class descrptors that follow (1)     ==24
    0x22, //report descriptor type (0x22 == HID)            ==25
    0x12,0x01, //length of report descriptor          ==26,27

  //endpoint descriptor
    USB_DESC_ENDPOINT_LEN, //length of descriptor              ==28
    USB_DESC_ENDPOINT_TYPE, //constant ENDPOINT (ENDPOINT 0x05)        ==29
    0x81, //endpoint number and direction (0x81 = EP1 IN)      ==30
    USB_ENDPOINT_TYPE_INTERRUPT, //transfer type supported (0x03 is interrupt)       ==31
    USB_EP1_TX_SIZE,0x00, //maximum packet size supported              ==32,33
    10,  //polling interval, in ms.  (cant be smaller than 10 for slow speed devices)    ==34

  //endpoint descriptor
    USB_DESC_ENDPOINT_LEN, //length of descriptor              ==35
    USB_DESC_ENDPOINT_TYPE, //constant ENDPOINT (ENDPOINT 0x05)        ==36
    0x02, //endpoint number and direction (0x81 = EP1 IN)      ==37
    USB_ENDPOINT_TYPE_INTERRUPT, //transfer type supported (0x03 is interrupt)       ==38
    USB_EP2_RX_SIZE,0x00, //maximum packet size supported              ==39,40
    100  //polling interval, in ms.  (cant be smaller than 10 for slow speed devices)    ==41
```

```
  };


  //****** BEGIN CONFIG DESCRIPTOR LOOKUP TABLES ********
  //since we can't make pointers to constants in certain pic16s, this is an offset table to find
  //  a specific descriptor in the above table.

  //NOTE: DO TO A LIMITATION OF THE CCS CODE, ALL HID INTERFACES MUST START AT 0 AND BE
SEQUENTIAL
  //     FOR EXAMPLE, IF YOU HAVE 2 HID INTERFACES THEY MUST BE INTERFACE 0 AND INTERFACE 1
  #define USB_NUM_HID_INTERFACES   1

  //the maximum number of interfaces seen on any config
  //for example, if config 1 has 1 interface and config 2 has 2 interfaces you must define this as 2
  #define USB_MAX_NUM_INTERFACES   1

  //define how many interfaces there are per config.  [0] is the first config, etc.
  const char USB_NUM_INTERFACES[USB_NUM_CONFIGURATIONS]={1};

  //define where to find class descriptors
  //first dimension is the config number
  //second dimension specifies which interface
  //last dimension specifies which class in this interface to get, but most will only have 1 class per
interface
  //if a class descriptor is not valid, set the value to 0xFFFF
  const int16
USB_CLASS_DESCRIPTORS[USB_NUM_CONFIGURATIONS][USB_NUM_HID_INTERFACES][1]=
  {
  //config 1
    //interface 0
      //class 1
      18
  };


  #if (sizeof(USB_CONFIG_DESC) != USB_TOTAL_CONFIG_LEN)
    #error USB_TOTAL_CONFIG_LEN not defined correctly
  #endif


////////////////////////////////////////////////////////////////
```

```
///
///   start device descriptors
///
//////////////////////////////////////////////////////////////////

  const char USB_DEVICE_DESC[] = {
    //starts of with device configuration. only one possible
      USB_DESC_DEVICE_LEN, //the length of this report   ==1
      0x01, //the constant DEVICE (DEVICE 0x01)  ==2
      0x10,0x01, //usb version in bcd (pic167xx is 1.1) ==3,4
      0x00, //class code ==5
      0x00, //subclass code ==6
      0x00, //protocol code ==7
      USB_MAX_EP0_PACKET_LENGTH, //max packet size for endpoint 0. (SLOW SPEED SPECIFIES 8) ==8
      0x1C,0x06,
      0x08,0x02, //product id   ==11,12 0x08, 0x00 original low speed
      0x00,0x63, //device release number  ==13,14
      0x02, //index of string description of manufacturer. therefore we point to string_1 array (see
below)  ==15
      0x01, //index of string descriptor of the product  ==16
      0x00, //index of string descriptor of serial number  ==17
      USB_NUM_CONFIGURATIONS  //number of possible configurations  ==18
  };

  #if (sizeof(USB_DEVICE_DESC) != USB_DESC_DEVICE_LEN)
    #error USB_DESC_DEVICE_LEN not defined correctly
  #endif


//////////////////////////////////////////////////////////////////
///
///   start string descriptors
///   String 0 is a special language string, and must be defined.  People in U.S.A. can leave this alone.
///
//////////////////////////////////////////////////////////////////

//the offset of the starting location of each string.  offset[0] is the start of string 0, offset[1] is the start
of string 1, etc.
const char USB_STRING_DESC_OFFSET[]={0,4,62};

//number of strings you have, including string 0.
#define USB_STRING_DESC_COUNT sizeof(USB_STRING_DESC_OFFSET)
```

```
char const USB_STRING_DESC[]={
  //string 0
      4, //length of string index                    // 0
      USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING) // 1
      0x09,0x04,   //Microsoft Defined for US-English      // 3
  //string 1
      58, //length of string index                   //  1  4
      USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING) //  2  5
      'A',0,                                 //  4  7
      'h',0,                                 //  6  9
      'm',0,                                  //  8 11
      'a',0,                                 // 10 13
      'd',0,                                 // 12 15
      ' ',0,                                // 14 17
      'W',0,                                  // 16 19
      'h',0,                                 // 18 21
      'e',0,                                 // 20 23
      'e',0,                                 // 22 25
      'l',0,                                // 24 27
      'S',0,                                 // 26 29
      't',0,                                 // 28 31
      'i',0,                                // 30 33
      'c',0,                                 // 32 35
      'k',0,                                 // 34 37
      ' ',0,                                // 36 39
      'V',0,                                  // 38 41
      'e',0,                                 // 40 43
      'r',0,                                 // 42 45
      's',0,                                 // 44 47
      'i',0,                                // 46 49
      'o',0,                                 // 48 51
      'n',0,                                 // 50 53
      ' ',0,                                // 52 55
      '1',0,                                 // 54 57
      '.',0,                                // 56 59
      '1',0,                                 // 58 61
  //string 2
      26, //length of string index                   //  1
      USB_DESC_STRING_TYPE, //descriptor type 0x03 (STRING) //  2
      'W',0,                                  //  4
      'h',0,                                 //  6
```

```
        'e',0,                          //  8
        'e',0,                          // 10
        'l',0,                          // 12
        'S',0,                          // 14
        't',0,                          // 16
        'i',0,                          // 18
        'c',0,                          // 20
        'k',0,                          // 22
        ' ',0,                          // 24
        'U',0,                          // 26
};
```

End of “main.h” ----------

Appendix E

Word cloud of this thesis