ICT313 Neuromend

Tempest

Design Document



# Title Page



**Project name:**

Neuromend

**Client/organisation:**

Shri Rai

**Supervisor:**

Fairuz Shiratuddin

**Tempest team members:**

Ary Bizar

Anopan Kandiah

Hannah Klinac

Alex Mlodawski

Bryan Yu

**Date of document:**

5/09/2014

**Version of document:**

1.0

# Contents Page



[Title Page 2](#_Toc397699849)

[Contents Page 3](#_Toc397699850)

[Executive Summary 4](#_Toc397699851)

[Introduction 5](#_Toc397699852)

[Data Design 7](#_Toc397699853)

[Process Design 8](#_Toc397699854)

[Architecture/Infrastructure Design 10](#_Toc397699855)

[Interface Design 13](#_Toc397699856)

[References 17](#_Toc397699857)

[Appendices 18](#_Toc397699858)

# Executive Summary



Purpose of the document is to provide a blueprint for building the system Neuromend and to provide a clear picture of the software. This document acts as a reference for the system, and will aid in the distribution of development among the development team. If the system is destroyed, it should be able to be rebuilt using this document.

This document is going to discuss data design, process design, architecture/infrastructure design, and interface design for the system.

# Introduction



The purpose of this document is to provide an outline for building the system Neuromend and to provide a clear picture of the software.

The intended audience for this document is the stakeholders associated with the project. This includes the client, supervisor and team members.

This system is the first designed and implemented product and it is called Neuromend.

Related documents:

* Requirements document
* Project management plan
* Final documentation

The above requirements document and project management plan are prerequisite documents, and provide background and context for this document and the system. Final documentation is the final document that results from this document as well as the prerequisites.

Size of the system to be implemented: 4 versions of the software simulation, each functioning with a different set of devices; Oculus Rift + Kinect, Oculus Rift + Leap Motion, Oculus Rift + Razor Hydra, Mouse + keyboard. The simulation has 3 levels. The simulation must have a networked database.

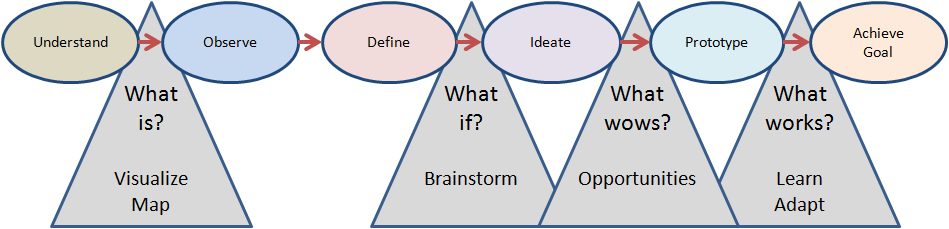
Complexity of the system: Each version of the simulation should be specific for the attached set of devices. The simulation should have a menu system. Each level should have a training stage and an execution stage. The time score for the execution stage should be stored on the user’s profile. The database is used to store user’s profiles with time scores for each level.

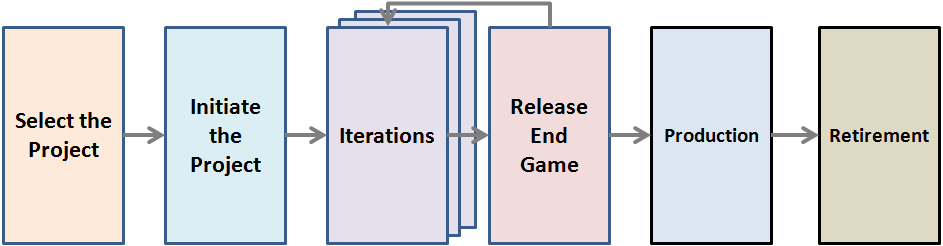
Arrangement of the design and implementation teams: 5 major areas of the system, each team member is responsible for one major area as follows:

* Level designer: Ary
* Oculus Rift + Kinect: Alex
* Oculus Rift + Leap Motion: Hannah
* Oculus Rift + Razor Hydra: Bryan
* Mouse + keyboard: Anopan

Chosen design methodology: Design Thinking. This methodology applies critical and creative thinking to understand, visualise, and develop approaches to solve the problem. Agile design is also used to break tasks into smaller increments of short time frames to deliver small working iterations to the system. The system itself has a design goal which uses design thinking, but agile design is used in smaller iterations to achieve that goal.

1. Define the problem: understand the problem, observe and understand to correctly define the right problem to be solved.
2. Create and consider options: come up with solutions to solve the problem.
3. Refine selected directions: adapt to dynamic conditions by prototyping.
4. Execute: achieve the designated goals.

Design thinking: 

Agile design:

# Data Design





# Process Design



**Process Descriptions**



Save Profile

1. Get user altered profile data
2. Request database to write changes to profile with new data

Load Profile

1. Make database request for all profile IDs
2. Display current profiles to user
3. Get profile selection from user
4. Request transfer of profile data from database with a given profile ID
5. Display profile information to user

Create Profile

1. Display form to collect initial user details
2. Request creation of profile from database

Delete Profile

1. Get profile id from user
2. Make request to database to delete profile with the given id

Request Performance Report

1. Set performance report selection criteria
2. Make database request for performance reports that meet selection criteria
3. Accept data from database
4. Transfer raw report data to **Format Performance** **Report** process

Format Performance Report

1. Read performance report data
2. Collate report data by date
3. Create graph with performance score extracted from each report data
4. Display performance graph to client

Connected Device Selection

1. Get list of connected devices
2. Get user selection of connected device
3. Request device data transfer to client from list
4. Transfer device data to **Configure Device** process

Configure Device

1. Retrieve changes to device settings from user
2. Write changes to device settings to device data

Game Level Selection

1. Get all game levels and display to user
2. Get level selection from user
3. If user accepts game level, transfer to **Play Game Level** process

Play Game Level

1. Retrieve level data
2. Transfer to game execution process with a level data

Process Game Level Summary Data

1. Read end of game data
2. Retrieve devices used, level ID, profile ID and current date
3. Request creation of new performance report from database with inputs from step **2)**

# Architecture/Infrastructure Design



**Architecture of the system**

Neuromend is simulation software for user interaction and manipulation of objects in a virtual world. It will be installed onto PC systems and work with several sensory devices in order to achieve this. Neuromend will be a fairly small program containing just three levels for users to play through. User performance will be tracked and stored onto a sever containing a database, which will be accessible anywhere with internet connectivity. There will be four versions of Neuromend, each supporting a specific device combination.

**Architectural diagram**

The diagram below is an example of how the system’s architecture will look like.

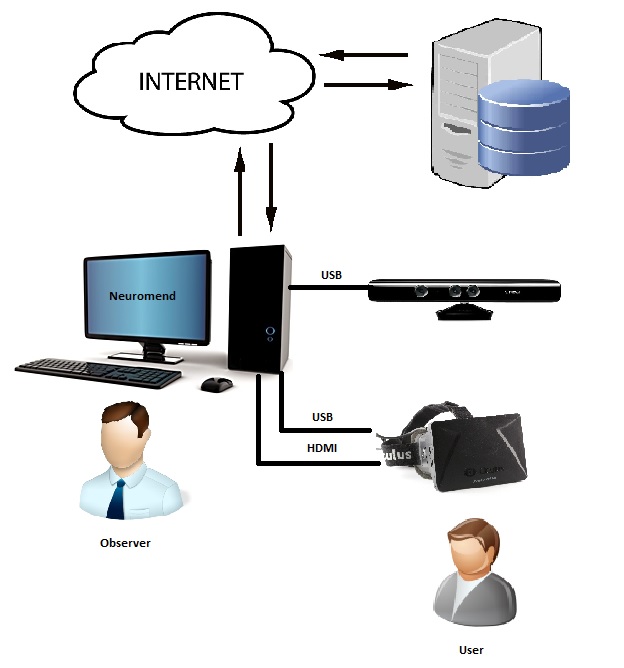


Figure 1.1 – Architecture example using Oculus Rift + Kinect combination [1].

**Additional software components**

In order for the architecture to be operational some outside software is required. These include the drivers for the sensory devices and their associated software development kits (SDK). The architecture makes heavy use of the SDKs provided with the sensory devices so they must exist on the system that will run Neuromend. All of this outside software will be included in the package.

**Infrastructure requirements:**

* **Capacity**

The system will work with one user at a time. It will have a user put on the Oculus Rift and operate their desired sensory device or mouse and keyboard. The sensory input devices include Kinect for Windows, Leap Motion and Razer Hydra. The program will have output shown on a monitor so others can observe what a user is doing. Installation of Neuromend needs to be done before operating the program so some available hard disk drive space will be required. Transactions will occur between the system running Neuromend and a server so a stable internet connection will also be required. These transactions involve sending and receiving users’ performance data.

* **Performance**

Systems that will run Neuromend will need to be relatively modern in order to smoothly display its graphics. This is important because slow/jerkiness will be disorientating for users, which will negatively impact their performances because their speed of each level completion is timed.

* **Integration & compatibility**

As mentioned in the performance section the system that will run Neuromend will need to be relatively modern. It will require a decent graphics card, internet connectivity, available hard disk space and USB ports. Compatibility between the sensory devices is required for operation. Systems will need compatibility with the following sensory device combinations Oculus Rift + Razer Hydra, Oculus Rift + Leap Motion, Oculus Rift + Kinect or a keyboard and mouse.

* **Platform strategy**

Systems will need to have Windows 7 to run Neuromend. Support and compatibility cannot be guaranteed on other versions of Windows or Macs because of the developmental nature of the sensory devices.

* **Security**

Neuromend will require users to input some personal data about themselves in order to accurately track their performances. This data will be stored on the system running Neuromend and a server communicated with through an internet connection. To make such data unidentifiable if there ever is a breach users’ will be identified using a unique ID instead of by names. Data stored in the database will be protected and have restricted access to those who have the right permissions. Access will be done through a log in system as a form of authentication.

* **Back-up & recovery**

Local failure on a system running Neuromend will not be a big issue because user data will still be stored on a sever so a recovery can be performed by getting lost user data back onto the system once it’s running again. However, there is a chance that the server may fail so local backs-up should continually be made by users. This should involve copying data over to external storage in case both their system and the server fail at the same time.

* **Scalability**

Design of Neuromend focuses on separation of functionality so it can be scaled up easily if required. This may include a multiplayer option so more than one user can play Neuromend at a time. It may also include using additional devices for finer control.

* **Future proofing**

The sensory devices developmental nature means that they will likely change quickly with time. Neuromend is not designed not to force users to upgrade devices should new ones become available. Updating of sensory devices may require patches for Neuromend to work with any new SDKs that accompany such devices.

**Alternative designs**

An alternative of designing four separate versions of Neuromend is to design just a single one that contains options in the software itself to switch between device combinations. However, this would prove fairly difficult because it was discovered early on that some of the sensory device drivers’ conflict with each other. This could cause system failures or device inoperability.

# Interface Design



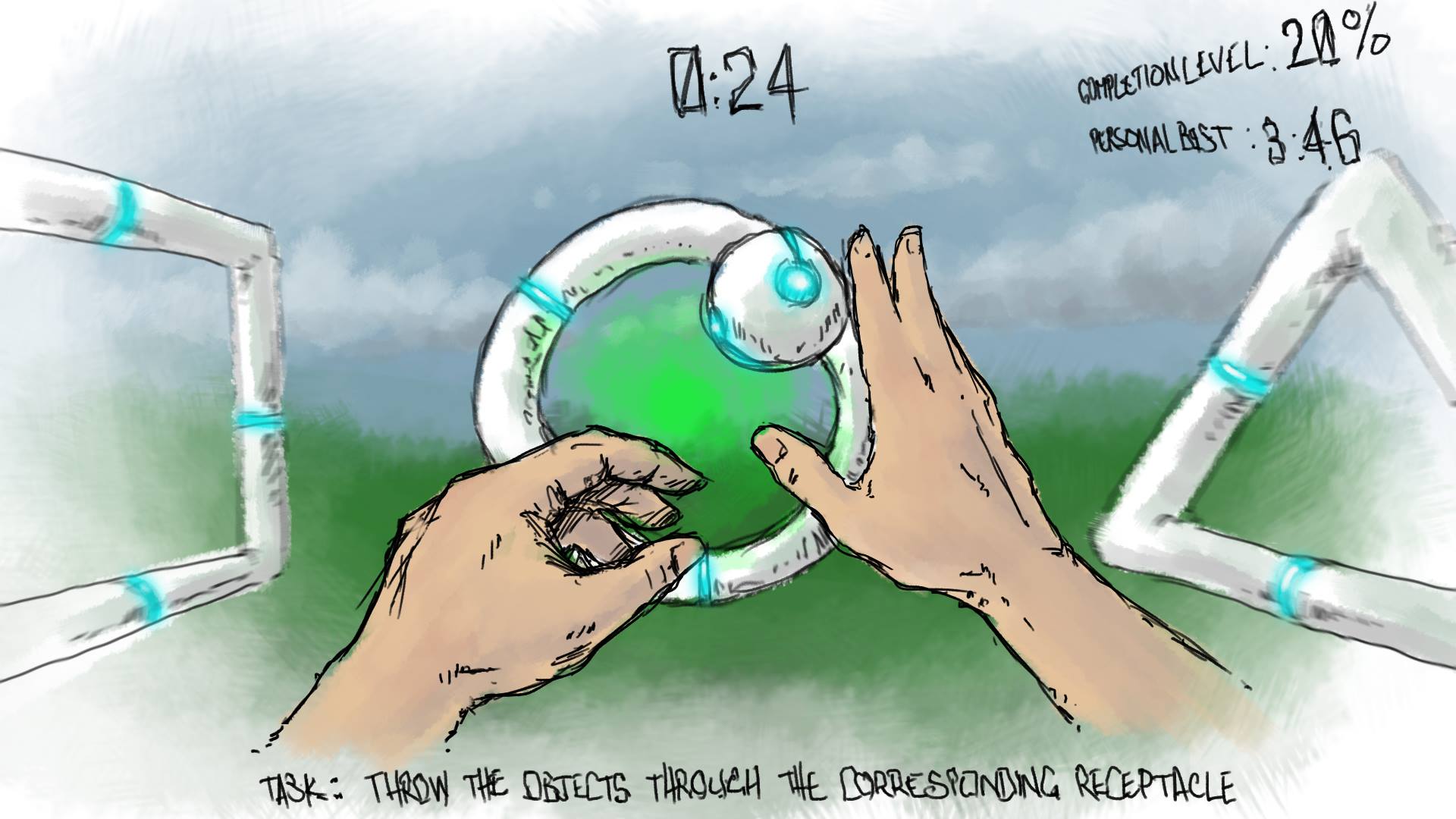
The design of the interfaces between software and non-human entities are almost the same for every selection of the devices. The drivers which are installed on the computers are the direct link between our program and the devices. Our program is then written to hand certain inputs from the devices as given by the device drivers. We program our interfaces according to what we need to perform in the simulation. Unity is what is being used to setup our programming interface and our main link between our code and the devices.

There is also data transferred between the system and the database which is externally accessible as well as accessible in the game. Back end: the system will communicate with the database using SQL. Front end: the information will be displayed using HTML formatting out of the game, and in the game it will be displayed on the computer screen in the game in text format. The data interfacing is still in design stage and is subject to change based on dynamic conditions.

The design of the interfaces between human and computer interaction is designed so that the user can get the easy feel of the environment as if they were there. In this project we have multiple different types of displaying our interface. Our interfaces change depending on our device interaction with the user. The user can choose between the uses of multiple devices to achieve their goal in the simulation. The idea is to test which selection of devices would be most suited to solve the problem. Each set of devices contains its own interface to bring the user into the simulation.

Inputs: Input is from the devices, and each device has different input methods as described below.

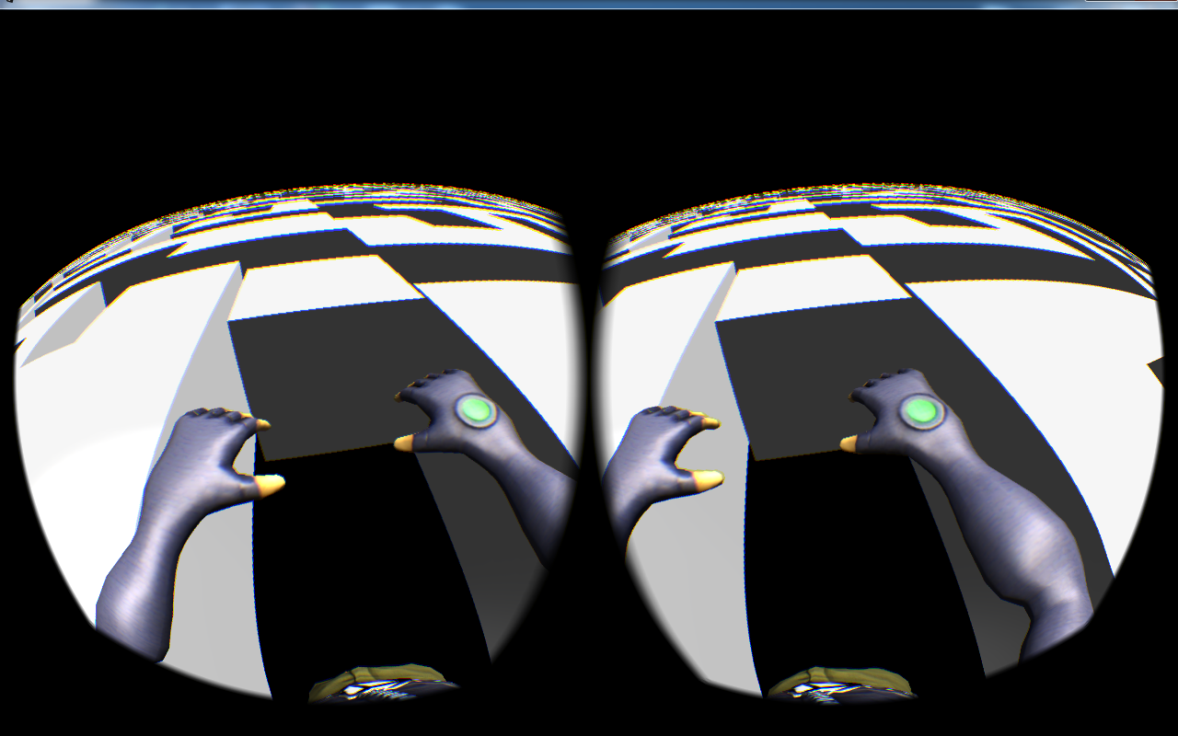
Outputs: Visual output via Oculus rift/computer monitor. Audio output from the computer. Data output to the database.

HUD interface design: 



Oculus Rift – Windows Kinect

The inputs for these devices are the motion and movement sensors of the Kinect and the head rotation and movement sensors of the Oculus. When the user has put the Oculus Rift headset on they can move their head to move their head in the simulation. If the user is standing in the line of sight of the Kinect the user may now move their own body parts with correct results in the simulation to the max extent of the Kinect capabilities.

Some of the finer inputs with the Kinect will be using the gesture control that is inbuilt into the Kinect sensor. We can use these gestures to give the user some other options as we do not want them walking around the room. We will make use of gestures to move the user forward when they pull with both hands and stop when they push. More gestures may be added in later for more control over the simulation. 

The Oculus movement follows through with all the combinations.

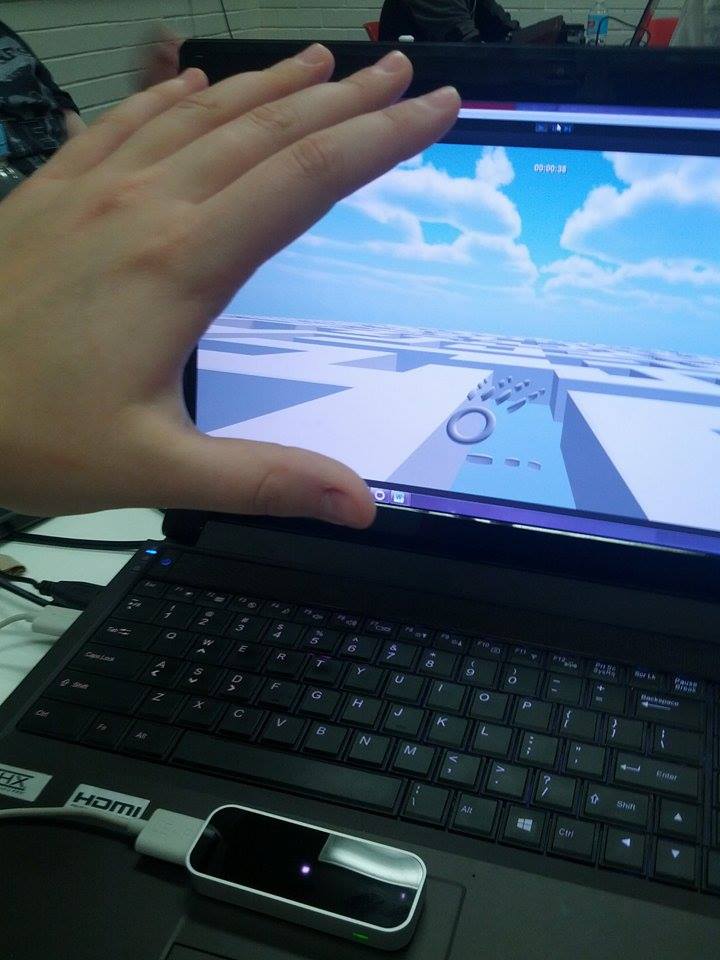
Oculus Rift – Razer Hydra

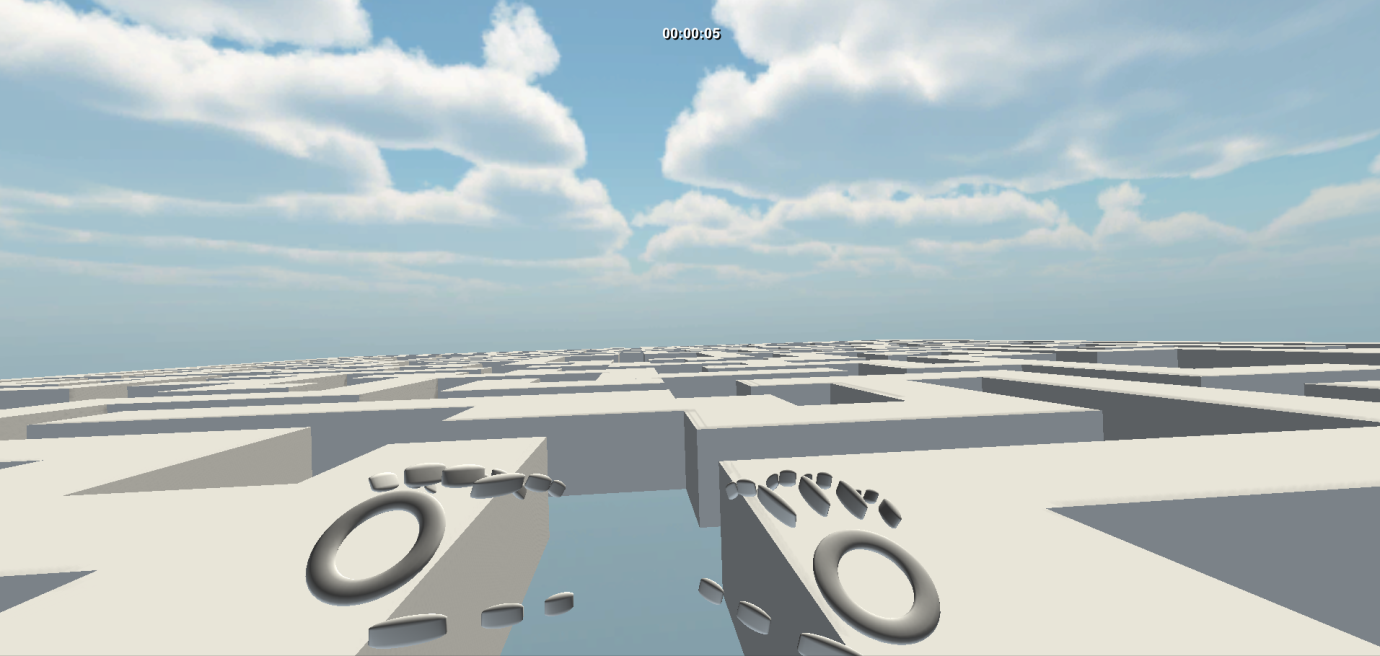
The Razer Hydra is controlled by holding the joysticks supplied. They have multiple buttons on then which allow the user to perform certain actions with their hands. Unlike the Kinect they do not have full control over every limb of their body and may only use their hands. The user can use the buttons on the side of the Razer Hydra to perform certain hand gestures. Left joystick on the Razer Hydra is for player movement. The right joystick is for camera movement (debugging purposes).



Oculus Rift – Leap Motion

The leap motion is a hand sensor so the user just has to put the Oculus Rift head set on and the leap motion should be attached and ready to receive hand movement information. Some gestures can be recorded to perform certain actions like moving the user forward, stopping and picking items up.





Oculus Rift – Keyboard/Mouse

This is mainly our testing input selection of devices. The keyboard and mouse is used for the times when a group member does not have access to a certain piece of equipment so this is used as a supplement.

# References

<http://www.nwlink.com/~donclark/design/design_models.html>



<http://www.rockwellautomation.com/resources/images/rockwellautomation/services/Generic_Computer--graphic_280w.jpg>

<http://www.dallmeier.ru/fileadmin/upload_electronic/Unternehmen/Niederlassungen/Dallmeier_Russland/Planning_icons_single/Shapes/JPG/Symbols/Cloud%20internet.jpg>

<http://www.advess.net/media/database_server1.jpg>

<http://www.blograzzi.net/wp-content/uploads/2011/04/arrow.jpg?234d14>

<http://www.tedesys.es/sites/default/files/kinect.jpg>

<http://upload.wikimedia.org/wikipedia/commons/a/ae/Oculus_Rift_-_Developer_Version_-_Front.jpg>

<http://www.softicons.com/social-media-icons/free-social-media-icons-by-aha-soft/user-icon>

<http://www.softicons.com/web-icons/user-icons-by-2shi/user-1-icon>

# Appendices



Appendix A: Deliverable Task Breakdown Statement

Appendix A:

