# Contents

1 Introduction 7

2 Related work 9
2.1 Motion Tracking 9
2.2 Techniques 9
   2.2.1 Active markers 10
   2.2.2 Passive markers 10
   2.2.3 Markerless systems 11
   2.2.4 Mechanical tracking 12
   2.2.5 Magnetical tracking 12
   2.2.6 Inertial tracking 12
   2.2.7 Head Tracking 13
2.3 The Wii remote 14
   2.3.1 Communication 15
   2.3.2 Motion Sensing 15
   2.3.3 IR tracking 16
   2.3.4 Interfacing the Wii Remote 17
2.4 3D viewing 18
2.5 Head Tracking with the Wii 19
2.6 Track IR 21
2.7 Summary 22

3 Head Tracking Library 23
3.1 Software libraries 23
3.2 Cross-platform 23
3.3 WiiUse 24
3.4 Open Source 24
3.5 Documentation 24
3.6 Sockets 25
3.7 TUIO Protocol 25
3.8 Calibration 26
3.9  Head Tracking Math ........................................ 26
3.10 Glasses .................................................. 28

4  Realizing the library ......................................... 31
   4.1  Library interface ........................................... 31
       4.1.1 Threading ............................................ 32
       4.1.2 Initialization ......................................... 32
       4.1.3 Configuration file .................................... 33
       4.1.4 Sequence ............................................. 33
   4.2  Point matching ........................................... 33
   4.3  TUIO Protocol implementation ............................ 36

5  A Desktop Virtual Reality Game ............................... 39
   5.1  The idea ................................................... 39
   5.2  Perspective ............................................... 40
   5.3  Field of view ............................................. 42
   5.4  Quaternions .............................................. 43
   5.5  OGRE3D ................................................... 43
   5.6  The Game .................................................. 44
   5.7  Collision detection ....................................... 46
   5.8  Projection Matrix ........................................ 46
   5.9  Evaluation ............................................... 47

6  Summary / Conclusion .......................................... 51

7  Future Work .................................................. 53

8  Bibliography .................................................. 55
Contents
Chapter 1

Introduction

Today’s virtual reality systems are relatively expensive. They either require a CAVE (Cave Automatic Virtual Environment) or a head mounted display. Such devices are not available for personal use. As computer games continues to get more realistic, Virtual reality could be the next level for realism in computer games.

Recently a graduate Ph.D student, Johnny Lee [16], demonstrated head tracking using the Wii Remote. The Wii Remote is the controller which ships with the Wii game console from Nintendo. When the Wii Remote was introduced it was a revolution in the game industry. Because its new idea of input was so radically different from the typical ways of interacting in a game, thus the Wii became instantly a huge success. What distinguishes the Wii Remote from traditional controllers is that it is equipped with accelerometers. These devices measure the acceleration the Wii Remote makes. Players can now interact by moving the Wii Remote, instead of pressing buttons on a controller. This means they can make natural movements, like the ones made in reality if playing games, like tennis or bowling.

Lee does not use these motion sensing technology at all. Apart from this the Wii Remote can also track infrared light sources. This tracking is normally used to track the position at the screen where the Wii Remote is pointing. This is done by tracking two sources of infrared on top or below the screen. This tracking system can also be used in other ways. Lee created a head tracking system by placing infrared light sources on glasses.

Inspired by the success of Johnny Lee this system was recreated for the project described in this thesis. A goal was to make virtual reality available for the average person. This can be done with a head tracking system. It can turn the monitor of a normal computer into a display with a correct perspective according to the position of the head. A Wii Remote is affordable for everyone, some people even already have one. The original application
was not reusable, the code of Lee had interleaved head tracking and rendering code. Developers wanting to use the same concept have to start all over. The software for this project had to be reusable for others who wanted to do a similar project, so the wheel does not have to be reinvented each time. This was another goal of the project. Finally we wanted to proof that the head tracking system works and is capable of creating a desktop virtual reality system.

For this project the system of Johnny Lee was recreated, but now in way it is easily reusable for others if they want to make a similar system. A software library has been created which can be used by other people in other projects. It is also made possible to tun the head tracking system as a separate application and to receive the head tracking data over a network connection. Finally to demonstrate the system in action a game has been created which uses the head tracking capabilities of our developped system. This game turns the player’s computer into a desktop virtual reality system. To make the game more interactive, the player has to actively move his head, for it to prevent losing the game.
Chapter 2

Related work

Head tracking is a special case of motion tracking. In the past there has already been done a lot of research into motion tracking. In this chapter the techniques and some related systems are described.

2.1 Motion Tracking

Motion Tracing is a technique to record movement digitally. It has several applications, the best known are motion tracking for capturing animations for movies and games. But there are also other applications like medical applications, for example a doctor can record the movements of a patient and use the recording for making a diagnosis for the patient. Other domains where analysing motion is useful are sports and dance [10].

2.2 Techniques

Motion Tracking can be done either direct or indirect. For example, the movements for a human character can be mapped directly from recorded movements of an actor. This is not necessary, it is also possible to use the motion of an arm and its fingers to control, for example, the motions of a face with its expressions. It can sometimes be easier to do this by hand, for cartoons for example the expressions of faces are exaggerated. Indirect can also used to animate animals in cartoons.

There are many methods to realize motion tracking. The most commonly used systems are optical systems, but there are also other ways to capture motions. Optical systems make use of multiple optical sensors (cameras) which are usually placed in a circle around the scene. The motion tracking software combines the input of these sensors and calculates the position and
orientation of the tracked object in space. Triangulation can be used to combine the input of multiple camera.

Most systems used for movies require the actor to wear a special suit. This suit is equipped with many special markers, which are easily identified in the image captured by the cameras. The markers send out light, so the software can concentrate on bright spots. Often infrared light is used, this is invisible for humans and does not interfere with regular light sources.

2.2.1 Active markers

There are two different kind of markers, there are active markers and passive markers. An example of active markers are shown in Figure 2.1. Active markers emit light themselves. They have as a disadvantage they have to be fed with power, so the actor has to carry a power source or he has to be connected with a wire. These requirements limits him somewhat in his freedom of movement. On the other hand, active markers can be seen from larger distances, which enables the use of larger recording sets. Another major advantage is that it is possible to flash each marker in a round robin fashion. In this way the software can identify markers much better. This is especially important in real-time systems, while in recordings for movies or games more time can be spend on algorithms that match the correct marker with an identified spot. In real-time this is not always possible, due to the limited processing speed.

2.2.2 Passive markers

Passive markers do not send light themselves. They reflect light from light sources placed around the scene. This way many more markers can be used and the actor does not need to carry electrical equipment around. Often
Fig. 2.2: Passive tracking is used to track finger position, made for a way of interacting with the computer [24].

Fig. 2.3: A markerless tracking system. The user does not have to wear any marker at all, tracking is done by computer vision, from [9].

all markers are equal, the software cannot distinguish them from each other in the image. The relative positions of the markers to each other can be a guidance for identification. The use of more cameras help with identifying the markers. For an example of a passive motion tracking system see Figure 2.2.

### 2.2.3 Markerless systems

Finally, there are also optical systems which do not use markers at all (Figure 2.3. They use computer vision to recognize where the actor is and in what position his body is. These systems are useful for large movements, but small precision movements are still difficult to track. At this moment these systems are still inferior to the marker systems.
2.2.4 Mechanical tracking

Besides the optical methods, there are also other methods available. It is possible to track motion mechanically (See Figure 2.4). These systems measure the angle of the body joints directly. The actor has to attach a skeleton-like suit to his body. There are sensors placed on the joints of the ‘bones’ in this skeleton that measure the angle of the body parts. This form of motion tracking is the cheapest way, it constrains the actor in its movement. It can be used to program a robot arm in a factory, but for realistic movie scenes it is not very suitable.

2.2.5 Magnetical tracking

Another approach is the use of magnetical tracking (Figure 2.5). A magnetic field is generated on the scene by three orthogonal coils. In each sensor also three coils are used to measure the magnetic flux. By measuring the relative voltage in each coil, the system can estimate the position and orientation of the sensor. The big advantage over optical systems is that sensors cannot be occluded. The system is very sensitive for metal.

2.2.6 Inertial tracking

The final approach to be mentioned here are inertial systems. They make use of inertial sensors, such as accelerometers to sense acceleration and movement of the body. The benefits for this method is the space that can be used is unlimited. There are no wires or external sensors necessary. This kind of system can be affordable as demonstrated by Vlasic in 2007 [25]. Unfortunately, accelerometers can only track relative movements, therefore, this method is usually combined with another technique.
2.2.7 Head Tracking

A special area of motion tracking is head tracking. By head tracking only a small portion of the body has to be tracked. Since a head has only one position (there are no joints in a head) only one position has to be tracked. This can significantly simplify the system for it can concentrate on one only point now. Virtual Reality is a common application of head tracking. The position of the head in space is essential in virtual reality. When a 3D world has to be simulated by 2D screens, information is required on the viewing position of the observer. According to studies head tracking can increase a task performance significantly [6].

In the center for Mathematics and Computer Science in Amsterdam a low cost desktop VR/AR system has been developed [18]. This system is called the Personal Space Station (PSS). For the correct perspective, head tracking is also included [19].

The head tracking system consists of two cameras for stereographic vision. The user of the system has to attach passive markers to his head. He has to wear glasses with three optical markers, which are easily identified by computer vision.

The first step of the process of tracking the head position is to recognize the markers in the image of the camera. The markers contain three black spots on a white background and are thus easily recognized. First, edges are detected in the image and after that there will be checked if the found spot is ellipsoid. If this is the case the spot is added to a list of found spots. These dots are combined with the input from the other cameras to reconstruct the head position.
2.3 The Wii remote

In late 2006 Nintendo released its newest game console, the Wii. This console was a revolution in the console world. The reason was the new way the player interacts with the console. Until the Wii all gaming consoles had a regular controller. The controller consisted of some buttons or joysticks. With the introduction of the Wii a new type of controller was introduced for gaming consoles, the Wii Remote (Figure 2.6), or just simply Wiimote. Its design is unconventional, it looks like a TV remote controller. What is so radically different on the Wii Remote is that the motion of the Wii Remote is used as the input for the game. Interacting with a game by moving the Wii Remote around, is a much more natural way than the interaction on conventional game controllers. Golfing or bowling on the Wii is much closer to reality then on other systems. Throwing a ball is no longer done by pressing a button, but is done by making a throwing motion with the Wiimote. Using the Wii remote is much more fun, but also a lot healthier, for some people who did not get their daily exercises before, get it now due to playing with the Wii Remote. Studies have shown that Wii players are better in doing very fine movements like surgery [21].
2.3.1 Communication

The connections between the Wii Remote and the Wii is wireless to support large movements without getting entangled in wires. The connection is realized with Bluetooth, a well-established protocol for short range wireless communication. The main application for Bluetooth is communication between handheld devices or for connecting wireless keyboards and mice to a computer. Bluetooth uses the 2.4 GHz band. It has a relatively low bandwidth if compared to other wireless protocols like Wifi. Because the Wii Remote uses this standard protocol for its communication it is really easy to connect to the Wii Remote from a regular personal computer. All that is needed is a Bluetooth dongle and it is possible to connect to the Wii Remote. This easy connection has opened the door for self-made appliances for the Wii Remote. For example a program was made that enables computer users to use a Wii Remote as a mouse pointer on the screen of their personal computer.

Apart from the hardware also a software implementation of the Bluetooth protocol is needed to establish a Bluetooth connection. This implementation is called the Bluetooth stack. For Linux this is easy, there is only one Bluetooth stack available, called BlueZ. Windows is a different story. For Windows there are a wide variety of stacks available. Only one stack can be used at the time. To switch to a different stack a whole new stack has to be installed to replace the other. Since Service Pack 2 of Windows XP, Microsoft included a Bluetooth stack in Windows. There are other Bluetooth stacks for Windows. A widespread Bluetooth stack is Bluesoleil. Many Bluetooth dongles ship with their own Bluetooth stack. Most of the Bluetooth stacks define their own APIs, therefore some application run only on some Bluetooth stacks. Also not all dongles are supported by all stacks, so there can be some compatibility problems in Bluetooth applications.

2.3.2 Motion Sensing

The Wii Remote has some buttons like a conventional controller. These buttons are used for certain actions, but movements are mainly done by actually moving the Wii Remote around. The Wii Remote is equipped with three accelerometers. This can detect the acceleration from three axes. Motion sensing is the technique used in most games. For example it is possible to slash a sword by just swaying the Wii Remote. A rigid body in 3D space has six degrees of freedom, as illustrated in Figure 2.7. It can move in any dimension (X, Y, Z) and it can rotate (pitch, roll and yaw). If the Wii Remote is in rest the acceleration is $g$ (standard gravity: $9.81 \text{ m/s}^2$) downwards, caused by the gravitational pull of the earth. Using this acceleration at three
axis the orientation and the acceleration of the Wii remote can be derived. The force which is sensed by the Wii Remote is:

\[ a = R(gz + ar) \]

Where \( a_r \) is the real acceleration and \( R \) is the rotation matrix. The rotation consists of three parts (roll, pitch and yaw), the acceleration also has three parts. There is only information available about the three components of \( a \), therefore this system cannot be solved. However with some assumptions it is possible to extract some useful information.

When the Wii Remote is at rest the only force it is experiencing is the gravity of the earth. From this we can easily extract the orientation (pitch and roll). Unfortunately, we cannot extract the yaw, for the gravity force does not change when rotating around the \( z \)-axis. However the Wiimote is equipped with an additional sensor which enables it to also get that information.

If the Wii is accelerating, it can be assumed that it does not rotate. In that case the previous found rotation is used. The value of \( R \) stays constant and we use this value to get the acceleration data. This simplification leads to the fact that acceleration and rotation cannot be detected at the same time. However, for games on the Wii this is not a problem.

Now that the acceleration is known, the speed of the Wii Remote can be derived as well. This can be done by integrating the acceleration. By integrating again we get the total distance traveled by the Wii Remote.

### 2.3.3 IR tracking

Besides acceleration the Wii Remote can also sense where it is pointing to at the screen. It uses a so called ‘sensor bar’ to get this information. The
sensor bar is equipped with two sources of infrared light. The name sensor bar is not really appropriate, it does not sense anything, it just sends infrared light. Sometimes other sources of infrared, like the sun or some lights, can interfere with the light sent from the sensor bar which forces player to close the curtains. Some inventive players have used two candles instead of the sensor bar, this also does the trick.

The Wii Remote is equipped with an infrared camera at the front. This camera works on a time resolution of 100 Hz, so it can sense fast motions very precisely. It has a spatial resolution of 1024 by 768 pixels, which is comparatively high for such a cheap piece of equipment as the the Wii Remote. A commercial camera with these specifications are not available for the same price.

As mentioned before, the camera is a high resolution camera and works incredibly fast (on 100 Hz). This generates a lot of data which has to be analyzed. It is way too much to send by Bluetooth to the Wii, for Bluetooth has a maximum capacity of 1 Mbit/s. Assumed that one pixels takes one byte, sending 100 images of 1024x768 pixels take 630 Mbit/s, which is way too much for Bluetooth to handle. The Wii supports four Wii Remotes connected on the same channel, limiting its bandwidth even more. To solve this problem the images are analyzed on the Wii Remote itself. The Wii Remote only sends the position and the size of the detected infrared dots. These are further analyzed on the Wii, which then calculates the position at the screen where the player is pointing. The Wii Remote analyses the screen with computer vision to extract blobs from the camera images. A blob is in computer vision terms a connected region in the image. The blob detection for the Wii Remote is fairly easy, due to the fact the perceived image is really clean. There is almost no noise which has to be taken into account. This is because the filter in front of the camera only lets infrared light through. Only the dots from the sensor bar are visible on the perceived image from the Wii Remote, unless of course there are other sources of infrared available. Sending the points from the processed image instead of the image itself saves a lot of bandwidth.

2.3.4 Interfacing the Wii Remote

When the Wii was just released, immediately hobbyist tried to reverse engineer the Wii. Especially the Wii Remote was interesting for it was an revolutionary new way to interact with a system. At this moment, the protocol which the Wii Remote uses to communicate to the Wii is almost completely known. The only thing which has not yet been discovered is the way the speaker on the Wii Remote works.
It was found out that the Wii Remote protocol is based on the Bluetooth HID (Human Interface Device) protocol, a protocol used for keyboards, mice and other input devices. The Bluetooth HID is essential the same as the USB HID [11]. A device which uses the HID protocol must be able to be used in a generic way. For example, a keyboard can have a lot of extra function keys. Although it as some extended functionality, it implements basic functionality of a keyboard, the normal keys. To support this way of communicating, when connecting to a HID device a description of the protocol is sent. It sent which kind of messages (reports in HID terms) it supports, and what the length of these messages are. This is also what the Wii Remote does. This description has helped much in identifying which reports are available for sending and receiving.

The discovery of how the protocol of the Wii Remote works has lead to the development of tools which can read data from the Wii Remote. There are several libraries written which can be used to control the Wii Remote. This saves the programmer a lot of work, when creating a custom Wii application. An example of a library for this purpose is Wiiuse [14]. Wiiuse is an open source library for using the Wii Remote’s functionality. It is very clean API, it uses non blocking and it is single threaded. For many applications it is important that the calls do not block, otherwise the whole application will come to a halt.

The WiiUse library is platform independent. It runs both under Linux and Windows. On Linux it uses the default Bluez Bluetooth stack. Under Windows it supports three different Bluetooth stacks. These are the native Windows Bluetooth stack, the Widcomm stack and the Bluesoleil stack.

### 2.4 3D viewing

Humans live in a 3D world, although most computer displays are flat. While it is possible to create a perspective image with computer graphics, the image is still flat. Humans can view the world in 3D, thanks to the fact they have two eyes which look at the world from a slightly different position. This way humans can estimate distances and speeds much better. Viewing from two different positions is called binocular viewing. With just one eye it would be impossible to estimate the size of an object. An object which is smaller can also be just further away. With two different point of views these two things can be distinguished. By correlating the two images from the two view points information about depth can be obtained. However the same cannot be done by an image on a computer screen. The image is rendered on a flat screen and both eyes see exactly the same, therefore there is no true 3D experience.
To make a display look 3D there has to be a different image for both eyes. There are several ways to do this. The first way to realise this is have a separated display for both eyes. In this case a special helmet (Head Mounted Display, HMD) is required. This helmet has two displays, one for each eye. It also possible to show the image of both view points on one screen. This screen is on a fixed position. The viewer needs special glasses which filters the left image for the left eye and the right image for the other eye. One way to do this is using horizontal and vertical polarized light. The screen should than be able to display these two kind of lights. This method is commonly used in 3D movie theatres, the visitors have to wear special glasses with polarized filters. When viewers tilt their head, the effect disappears for the polarization direction of the screen and glasses are not aligned anymore. Another way is to use two different colors for each eye, the viewer then needs color filters in his glasses. The big advantage of this is that regular displays and printed work can show these images. Since two different colors are required for each eye only monochromatic images are possible with this technique.

For a correct view the position of the observer should be known, otherwise people can only see a correct perspective when they are on a fixed predefined spot. With the help of head tracking it is possible to get the position of the eyes and display the correct perspective. Displays using this method are called Head-tracked display (HTD). Now the screen turns in to a hologram and the perspective changes when looking from a different angle to the screen. Usually this displays can only be used by one person at the time, but there is research to displays for more than one user. [23] Some displays send different views in different directions. In this case multiple users can use the display, it also has as another advantage that nothing is required on the user’s head.[8]

2.5 Head Tracking with the Wii

Johnny Lee[15] is a scientist at the Carnegie Mellon University in Pittsberg. He has a number of papers on his name. On the internet he is well known for his projects with the Wii remote[16]. He has used the Wii Remote for some spectacular things, where the Wii Remote was never intended for. He has created a finger tracking system, a multi-point interactive whiteboard and a head tracking system. All of these project use the Wii Remote ‘reversed’. Instead of a stationary source of infrared and a moving Wii Remote, Johnny Lee position his Wii mote at a fixed position. The infrared sources are placed on the moving objects. In the case of finger tracking the fingers reflect infrared send by an array of infrared LED’s. The multitouch whiteboard
uses pens which have a infrared LED at the tip of the pen.

Finally Johnny Lee has created a desktop virtual reality system using the Wii Remote. With a special set of glasses with two LED’s on top of it, it is possible to track the position of the head in space. This position can be used to create a 3D image on the screen which is rendered from the actual viewpoint of the user of the system. This way a far more realistic 3D experience can be created, which reacts on movement and the changes of the position of view of the observer. It looks like the screen of the computer is turned in a window to a virtual world. It is possible to look around the corner and when moving closer to the screen a bigger overview of the scene is visible.

Johnny Lee has has created a small demonstration program which implements a simple desktop virtual reality. The program shows a room which has the same size of the screen. In this room there are floating targets, some of them actually appear to float out of the screen, so strong is the illusion by creating a perspective from the point of view of the viewer. The system works only for one person at the time, if it would be possible to track two glasses at the same time, still one perspective can be rendered on the screen.

The demoprogram of Johnny Lee can only track three of the six degrees of freedom. It is only capable of tracking translation, but rotation is not implemented. For virtual reality this is not necessary, because the view does not change for a user when his head is rotated. When rotated the angle from the Wii Remote to the head remains the same. It would be impossible to track all three possible rotations. Roll is possible to track, when the head is rolled one dot appears higher than the other. But when yawing from the left to the right, the only visible change is that the dots get closer to each other, which can not be distinguished from moving away or getting closer to the screen. Also a change in pitch cannot be distinguished from a movement along the Y axis.

Three months after Johnny Lee first showed his head tracking system, Electronic Arts announced to include head tracking in a game for the Wii. The puzzle game Boom Blox would include head tracking using the method of Johnny Lee as an easteregg. Boom Blox would be first game on the Wii featuring head tracking. Unfortunately Electronic arts has decided to drop the head tracking functionality in Boom Blox. The publisher has not made a statement why this function was dropped.
2.6 Track IR

The same principle Johnny Lee uses with his head tracking system is also used in commercial head tracking systems. An example of this is Track IR\cite{5} of the firm Natural Point. Natural Point is specialized in motion tracking, besides track IR it also sells full body motion tracking equipment, like reflectors, cameras and infrared light sources.

Like Johnny’s system, track IR comes with a camera to track infrared light dots and a pair of glasses (Figure 2.8) with infrared LED’s. Instead of two LED’s the glasses of Track IR uses another third LED. This enables the system to capture all six degrees of freedom. Not only translation is captured but also rotation. Games that uses Track IR, use the rotation to rotate the screen, so that you can look around in your environment. Although this is of course really cool, it is not really realistic. When looking through an ordinary window, the room behind the window does not roll when the viewer is rolling his head. In games which support the use of Track IR this leads to panning the screen. This is also not reality, you have to move your eyes and look from the corners of your eyes to see the screen when you have yawed your head to a direction. In real world you would expect to be able to look straight ahead, but unfortunately the screen is stationary and stays on the same place. Johnny Lee’s method does not suffer from this problem, for it uses another concepts. In the case of Johnny Lee’s demo you could regard your screen as a window to another world. With the Track IR system this is not the case, it uses the head position as a position for the in game camera and the orientation for the orientation of the camera.
2.7 Summary

For motion capturing there are a wide variety of techniques available. Extensive research has been done to all of them.

The Wii Remote, shipped with the popular Wii game console, has in-built infrared tracking capability. The performance of the camera is very good for its price class. Because the Bluetooth protocol the Wii Remote uses is easy to understand and to reverse engineer and also because the WiiMote is so innovative, many people have made their own projects using the Wii Remote. One of them is Johnny Lee, he has used the tracking capabilities to create his own head tracking system.
Chapter 3

Head Tracking Library

For this project I wanted to create a head tracking system using the Wii Remote, like Johnny Lee has demonstrated. Lee made a small demonstration program in C#. His code is just a proof of concept, it is not useful for reusing in other applications. The head tracking code Johnny Lee wrote is interleaved with rendering code of the 3D scene. This makes it really difficult to reuse the code for another head tracking application. C# limits the user to the use of Microsoft .NET. I wanted to create a head tracking program which can be reused as a component in other applications by other developers who want to use head tracking in their project.

3.1 Software libraries

A way to make reusable software is by putting this software into a software library. Other applications can use the code contained in the library. A library can be used by different applications which all use the same routines from the library, it is independent from the application where it runs in. The communication between the application and library is done by API’s (Application Programming Interface). The API’s form the interface to the library. In the API a set of routines is defined what can be used by code outside the library. The application using the library can use these routines to use the functionality of the library.

3.2 Cross-platform

A demand for the library was that it had to be cross platform. People have to be able to use the library on as many platforms as possible, so that the library does not restrict the user of it in their choice for a platform. For
developing the library a programming language had to be chosen which could realize this requirement. With Java it is possible to create a library which can be run on all platforms without any adaptations. The reason there is not chosen for Java is that it will then limit the programmer to use the Java programming language. Another reason not to choose for Java is the poor implementation of Bluetooth in Java.

For the tracking library C is chosen as a programming language. C is one of the most popular programming languages. A program written in C can be compiled to almost any platform, if no platform specific API's are used in the program. Because C compiles to native machine code it creates really fast code, for the language is designed so that it can be easily mapped to machine instructions.

3.3 WiiUse

For C there has already been put a lot of effort in making a library for the Wii Remote. For the head library it is unnecessary to reinvent the wheel if there is already put much work in creating a library for connecting to the Wii Remote. That is why an existing Wii library will be used for this project.

For this project the library Wiiuse is chosen. WiiUse is a really interesting library. It has a light weight API, it is cross platform and open source. Wiiuse supports both Windows and Linux, so using this library means that the head tracking library can be kept platform independent. Unfortunately a Macintosh is not supported by WiiUse, this means this platform is also ruled out by the head tracking library. For the rest the library fits in really well in what is needed for the project. By using this library the focus can be put more on the head tracking code than on interfacing with the Wii Remote.

3.4 Open Source

The library shall be open source, so that other people who want to work on the library can easily adapt things to their liking. If there are still any bugs present, they can be patched by the users themselves.

3.5 Documentation

For the users of our library and also for people who are working with the source it is important they can understand the code. The source code should be commented very well so it is very clear for users what happens in the
code. For the library users there should be documentation available where they can find what library calls are available to use.

It is possible to combine those two requirements. By including special tags in the comments through the source code, documentation can be automatically generated. When writing comments through the code the documentation gets written by itself. For the head tracking library Doxygen is used to generate the documentation from the source code comments. Doxygen can generate comments from the source code in both HTML format and in Latex, so also a PDF can be created.

3.6 Sockets

Since C library is used, it is required that application using the library can import and link to a C library. This is not possible in all programming languages, or in some it is very difficult. For example in C# all library calls have to be wrapped using the so called “platform invoke”, in Java the Java Native Interface is needed for this purpose.

To overcome this and make the library accessible to all programming languages universal interface using network sockets is created. This has an extra advantage the head tracking code can be run on a separate machine. If the head tracking system crashes for some reason, the application can keep running normally.

The de facto standard for exchanging messages over a network is SOAP. SOAP however is a really verbose protocol, for it is based on XML, which has a long syntax. The messages which are build in XML are then send by some other transportation protocol like HTTP, which adds even more overhead. Since the Wii Remote sends data at a rate of 100 Hz, this is too much for SOAP to handle. A better protocol had to be found.

3.7 TUIO Protocol

The reactable [12] is collaborative electronic music instrument with a tabletop tangible multi-touch interface. For this device a protocol was developed which defines common properties of objects on the table surface and also for finger and hand gestures performed by the user, the TUIO protocol. The demands for those gestures are similar to those for head tracking. The TUIO protocol is therefore a good option for the head tracking project.

The protocol is based on UDP. UDP is a high performance network protocol. There is no overhead added to ensure that packets arrive and that
they arrive in correct order. However it is not much of a problem for head tracking if one packet misses in a stream. It just makes the movement a little less smooth, but probably the user does not even notice this. The same story applies for packets arriving in swapped order. The TUIO protocol is derived from the Open Sound Control (OCS) protocol [26]. This protocol was designed for communication among computers, sound synthesizers and other multimedia devices. In OSC data is sent in packets. A packet can contain either a single message or a bundle of messages, to make the most efficient use of a single packet.

Because a tracking library is already implemented the network version can be a program which uses the library made. The only thing this program has to do is receive data from the library and set it over the network to the host which is interested in head tracking data.

### 3.8 Calibration

The tracking library gets from the Wii Remote the position of infrared LED sources in his view. It has a two dimensional view of 1024 by 768 pixels. It also gives an estimate of the size of the detected infrared blob, this is given at a scale from 0 to 15. The task of the library is converting these data in the 3D position of the head. It thereby needs some parameters which are variable and are put in a configuration file. One of these variables is the vertical angle the Wii Remote is in. The position of the head is relative to the center of the screen, therefore the library should also know the position of the Wii Remote relative to the screen. These are the other parameters.

The library must be able to read these parameters from a configuration file. The angle can be calibrated by a special calibration program, which needs to write data back to the configuration file. To make sure that the configuration file does not get corrupted by programs writing the settings in a wrong way the library should support writing settings back to the file. Now the library makes sure that the configuration file remains in the correct format.

The architecture of the library and head tracking applications can be summarized as in the scheme in picture 3.1.

### 3.9 Head Tracking Math

From the Wii Remote two points in a two dimensional plane are received. Using these two points an estimation for the head position in 3D space has to be made. For the distance in the Z direction the distance between the two
CHAPTER 3. HEAD TRACKING LIBRARY

![Diagram of head tracking library](image)

Fig. 3.1: The architecture of head tracking applications using the head tracking library

dots is used. The distance between the dots on the glasses is known as well is the viewing angle of the Wii Remote, this is 45 degrees. When the perceived dots span about 60 percent of the view of the Wii Remote it assumed that this is 60 percent of the 45 degrees of the viewing angle. The distortion of the camera is ignored. Knowing the angle between the Wii Mote and the dots, the tangens can be used to calculate the actual distance to the Wii Remote, this is illustrated in Figure 3.2.

The following formula is used to calculate to the WiiRemote. Hereby is \( z \) the distance to the Wii Remote, \( d \) is the distance between the two LED’s and \( a \) is the perceived distance between the dots (where \( a = 1 \) means one dot is on the left edge and the other is on the right edge).

\[
z = \frac{d \times 0.5}{\tan(45\degree \times 0.5 \times a)}
\]

Because the center of the two points corresponds to the center of the head, this center can be used to calculate how far the glasses have moved to the left or the right. This center point can be mapped to an angle at the same way the angle between both infrared sources is estimated. The viewing angle is linearly interpolated, if the center lies on the edge of the view, this corresponds to an angle of 22.5 degrees. Using the calculated distance of the glasses the sine function can be used to calculate the \( x \) coordinate. This
Fig. 3.2: Calculating the distance to the Wii Remote

leads to the following formula:

\[ x = z \times \sin a \]

Hereby is \( z \) the distance to the Wii Remote and \( a \) the angle to the center point. The calculation is illustrated in Figure 3.3.

### 3.10 Glasses

For realizing head tracking not only software is required, but also some hardware. The motions of the head is captured by the camera in the Wii Remote. The Wii Remote only captures infrared dots, so the head has to be equipped with infrared light sources which acts as active markers for the head tracking system. In the hardware store cheap safety glasses which have two LED’s mounted on top of them are available. These LED’s send out normal light, which lies in the visible spectrum. They have to be replaced with infrared LED’s to make the system work.

In Figure 3.4 the glasses made and used for head tracking are displayed. In the left image the LED’s are switched off and at the right they are turned on. The camera used to take can sense infrared light, for the human eye it is invisible. With the naked eye it is impossible to see whether the glasses are turned on or off.
CHAPTER 3. HEAD TRACKING LIBRARY

Fig. 3.3: Calculating the horizontal position of the head

Fig. 3.4: The glasses used for the head tracking
Chapter 4

Realizing the library

For the realization of the library we started working on a Linux version and later we ported it to Windows. From that point we maintained two versions and put effort that it would compile under both operating systems.

4.1 Library interface

Head tracking applications are interested in the position of the head, they want to receive an update of this data as many times as it is possible for the Wii Remote to supply this information. This is about 100 times per second. It is logical for the library to raise an event so the application can respond as soon as there is tracking information available. Therefore it is made possible to register an event handler in the library. To give programmers more freedom in the way they use the library another call is added to retrieve the last head position directly. This can be useful if the head tracking application is not an event driven application, but needs the head tracking information always at a certain moment. This is the case in most computer games, which consists of a main loop in where all important things are updated, like the screen and the position of objects. As a part of this loop the head position from the head tracking library can be retrieved.

The head position is a 3D point in space, a representation of this position had to be found. The representation chosen for this project is a 3D coordinate. All distances are in meters. The origin is the center of the screen, which lies on position (0, 0, 0). We chose z to be positive, when the head is further away from the screen it has a higher value. Another possible representations of the head position is using the angles under which the head is visible together with a distance. This seems an obvious choice, for the perspective depends also on these parameters. Because there is a transformation
necassary from the position of the Wii Remote to the position of the screen, this is not a useful representation. For creating the correct perspective the first representation is also better, for the camera can be placed on the head’s position.

An extra data field is used which indicates whether the head is visible, it is possible that no or just one infrared dot is visible to the Wii Remote in this case the present field is set to false. The data structure of the head position is defined as following:

```c
struct HEADPOINT {
    bool present;
    float x;
    float y;
    float z;
};
```

### 4.1.1 Threading

To get data from the Wii Use library it has to be polled regularly. This also requires the head tracking library to have a polling function, which delegates control to the Wii Use polling function. After calling the polling function the registered event handlers are called with the head position. This implementation is not truly event driven, it forces the head tracking application to poll the library regular. To overcome this another way is included where event handlers are called asynchronously. This is implemented by creating a new thread inside the library which keeps polling Wii Use in a regular interval. This interval time is configurable with a library call or in the configuration file.

### 4.1.2 Initialization

Initialization of the Wii Remote is done in three steps. In the The first step the library is initialized as a whole. It is possible to supply an event handler at that moment. This is where the event is handled in the case the Wiimote is disconnected. This call also initializes the Wii Use library. The next call is to connect connect to the Wii Mote. This is done in a separate call, so it is possible to delay connecting if necassary for the application. After this connection is established head tracking has to be enabled. With this third there is also the possibility to supply an event handler. This is the event handler where head tracking data is send to. By supplying NULL, no event handler is installed and the head position has to be retrieved manually.
Head tracking is enabled with its own call because the library also supports finger tracking, for another project. In the case finger tracking is needed instead a different call has to be use. By enabling head tracking the Wiimote infrared tracking is automatically turned on as well by the library. The Wii Remote does not enable this by by default, because the processing of the camera image takes a lot of CPU and thus also a lot of battery power. To reduce power usage the Wii Remote can disable tracking for games which only supports motion sensing.

4.1.3 Configuration file

Optional a configuration file can be supplied to the library. This file contains the calibrations parameters needed for the library. These are the relative position of the Wii Remote to the center of the screen, the distance between the two LEDs on the glasses and the angle the Wii Remote is placed in. When no configuration file is supplied, hardcoded defaults will be used for these parameters. By default it is assumed the Wii Remote is put 12 centimeters below the center of the screen and it is aligned perpendicular to the screen. When no value is supplied for the LED distance, 15.5 centimeters is used. This is the actual LED distance for the glasses created for this project. It is also possible to override each parameter manually with a library call, this is useful for the calibration program. It can update all parameters and then use another library call to write the configuration back to the configuration file.

4.1.4 Sequence

A typical sequence of functions calls for a head tracking application is displayed in the sequence diagram in Figure 4.1. The polling part can be extended for as long as the application runs. In Figure 4.2 an example sequence diagram is displayed for an application which uses an apart thread to poll the Wii Remote.

4.2 Point matching

The glasses only have two infrared LEDs, but the Wii Remote can track up to four points. Occassionally it happens that more points show up. This can happen due to small reflections of the LEDs in the plastic of the glasses. When this occurs another small dot appears in the vicinity of one of the other points. When just looking naively at the first two points it is possible to take
accidently the two closest points as the positions for the LEDs. Because the points are very close the tracked position is very far away, for about two or three meters from the Wii Remote. When this happens for a head tracked display a totally different view will suddenly be displayed. This is disorientating and annoying. To prevent this from happening a way is required to detect reflected points.

When more than two points are visible, there have to be some false points. Probably a reflection just appeared. By comparing all points with earlier detected points it is possible to find the points which are most likely to be false positives. Those two points are let through which are closest to the two detected points in the previous frame. The algorithm is greedy, first it picks the point closest to point one from the previous frame. After that it picks the point closest to the second all remaining points. This trick improves the head tracking system a lot, when a third point pops up the position of the head remains stable.
Fig. 4.2: A sequence of function calls when using a separate thread for polling the Wii Remote.
4.3 TUIO Protocol implementation

In TUIO there are three kind of messages, the alive, the set and the fseq message. Since TUIO is designed to be able to handle more objects at the same time, each object has its own identifier. TUIO distinguishes two kind of identifiers, the session id and the class id. The session id is unique for each object, while the class id is the same for each object of the same kind. An alive messages contains a list of points which are visible. For the head tracking application there is a maximum of one alive object at the time, because there is only one head. So the alive ‘list’ has a maximum of one identifier or is empty if the head is not present. The head is always identified with session id and class id 1.

The set messages contain the actual position data. The TUIO protocol specifies several profiles. A profile defines what parameters are tracked. For example the profile ‘2D interactive surface’, has parameters for the position, the angle, the movement speed and direction, the rotation speed and direction, motion acceleration and finally rotation acceleration. There is a similar profile for a 3D object. The only thing relevant to send for head tracking is the position, any of the other parameters are not needed. Some of them cannot not even be known, for the head tracking system has only three degrees of freedom. TUIO does not have a profile with only a position. Fortunatly the TUIO protocol supports user defined profiles. To define a custom profile, the profile format has to be send instead of the profile name. The profile format used in this project is ‘isxyz’. This means that the set messages have the following parameters: i for class identifier, s for session identifier and the x, y, z are the head position.

The last message type is the fseq message. It has an attached frame sequence number which keeps increasing. TUIO uses UDP and can therefore not guarantee the order in which the packets arrive. This sequence number can be used to reconstruct the order of the packets. Each bundle of messages should contain a message of this type.

Because Open Sound Control is quite complex to implement an existing implementation in C++ is used [7]. It uses operator overloading in C++ to build messages. This makes the syntax for building a message clean. For example this is the way a set message is created:

```
<< osc::BeginMessage(STR_Tuio_profile)
<< "set"
<< 1
<< 1
<< x
```
CHAPTER 4. REALIZING THE LIBRARY

<< y
<< z
<< osc::EndMessage
Chapter 5

A Desktop Virtual Reality Game

In the previous chapters it is described how a head tracking system was created. This system can estimate the coordinates of the head in space. Those coordinates alone are not very interesting, so for the second part of the project an application was created, which shows the head tracking library in action. There are a few different applications for head tracking. With head tracking the user’s head can be used as an input device. This is useful for people who are disabled and who cannot use the mouse. The main application of head tracking is however virtual reality. With head tracking it is possible to make a head tracked display. This means that the projection on the screen is adjusted to the viewpoint of the user.

5.1 The idea

The first idea for the application was to use head tracking under a normal Windows like desktop environment. By moving the head from left to right windows be viewed from a different perspective. If the user wants to know what window is behind another, instead of minimizing the window on top he can just look behind it. The effect created can be very nice, but it is not very useful. Also it has already been created as a plugin for Compiz [3]. Compiz is a 3D desktop environment for Linux, comparable to the Aero look in Windows Vista.

Because head tracking is mainly used in virtual reality application, it was an obvious choice to make such an application with the library created. There are not many games available in virtual reality for desktop computers. That is why a game was created for this project as a head tracking application.
By enabling head tracking for a game, the game experience will get a lot more realistic. Although it is possible to display a 3D scene on a 2D screen, the image will not be truly 3 dimensional. Looking from a different angle does not give you another view of the scene, this is just like a picture or a painting. With an ordinary window with another room on the other side this is different. Looking from different angles through the window will get different views for the observer. For the game developed it was a goal to recreate the effect of a normal window. This way the screen of the computer turns into a virtual window into the virtual world.

Another thing that had to be added to the game is that head movements also had a meaning in the game. The head position is already tracked for rendering from the correct perspective. Since a game is made and this data available, this input can also be used in the game play. This way the player has more reason to change his head position, other then getting a different view of the game area.

Because a virtual window is created a way to use this concept in a game is by creating a shooter. The player shoots through the window on targets which are on the other side of the window. Projectiles are coming for the player, so the player has to dodge these missiles by moving his head to the side, or by ducking in which case the missile will fly over his head.

5.2 Perspective

In 3D games usually a perspective projection is used for rendering the scene. This means there is a fixed viewing angle from a certain position in the scene in a certain direction. Generally the viewing angle is about 45 degrees. Due to this perspective, objects which are further away appear smaller on the screen. This projection is created by a few parameters. These are the camera position, the viewing direction, the horizontal and vertical viewing angle and finally an up vector which indicates the roll of the camera. Finally there is a near plane and a far plane which defines at what distances objects still get rendered. A perspective projection creates a frustum sized view. 5.1.

For all normal desktop renderings the projection plane is always perpendicular to the viewing direction. When taking the head position into account when creating the perspective, this will not always be case. The player of the game can have his head shifted to the left, then the angle the viewing direction makes with the screen is not equal to 90 degrees as illustrated in Figure ??.

If the head is not in the center of the screen, the projection frustum will get sheared. When moved to the left the frustum gets sheared to the right
CHAPTER 5. A DESKTOP VIRTUAL REALITY GAME

Fig. 5.1: Perspective projection. Near and far plane are called front and back plane respectively. Source: http://www.csee.umbc.edu/rheingan/435/pages/res/gen-8.Viewing-single-page-0.html

Fig. 5.2: On a head tracked display the projection plane is only perpendicular when the head is in the center of the screen.
Fig. 5.3: An asymmetric projection. Source: www.vis-sim.com/performer/pf_faq_1.htm

Fig. 5.4: The field of view gets larger when getting closer to the screen. In the left image the head is moved away from the screen, in the right it has moved away from the screen.

(see Figure 5.3). Due to the sheering the observer gets to see more of the right. This kind of projection is called ‘off-axis projection’ or ‘asymmetric projection’.

### 5.3 Field of view

Another parameter which changes when the head moves to another position is the field of view, or viewing angle. Normally this is a fixed value, but when we regard the screen as a window the viewing angle should change when moving away from or getting closer to the screen. This is illustrated in Figure 5.4.
Fig. 5.5: In the top Figure you can see it is possible to rotate along all the axis. After a serie of rotations you can get in a situation like the bottom Figure. Note there is no rotation axis left to perform the rotation indicated by the arrow. Source: http://history.nasa.gov/ap08fj/10day3_maroon.htm

5.4 Quaternions

At several points in the game rotations are used. To represent a rotation ‘quaternions’ are used.

There are several possible ways to represent rotations in three dimensions. The first is with euler angles. These are three angles ($\alpha$, $\beta$, $\gamma$) which represent rotations over the x, y and z axis, executed in that order.

The big problem with this method it suffers from an effect called ‘Gimbal lock’. This happens when two axes of rotation become aligned and one of them has no longer effect. Gimbal lock is illustrated in Figure 5.5. Another problem with Euler angles is that it is very difficult to find the necassary rotation to face in a certain direction.

A quaternion is essential the same concept as a complex number, but then taken to the fourth dimension. A quaternion is in the form $a + bi + cj + dk$. Hereby is $a$ the angle over which is rotated and $(b, c, d)$ form a vector on where around the rotation takes place.

5.5 OGRE3D

To realize the game the first decision to made was how to create the graphics. Creating the game directly with the OpenGL or DirectX graphics library would be too much work. These libraries support only primitive graphic operations. For making a game, a game engine can be used to program at a much higher level. A game engine does the rendering, the user of the
engine only needs to specify the objects and the scene and does not have to concern with basic graphic operation.

The choice for the game fell on OGRE3D. OGRE3D is an 3D rendering engine [4]. It is a open source library and it is also cross platform. Besides Windows it also runs under Linux and Mac OS. The demo is never tested on other operating systems than Windows, but porting it to Linux should be easy. OGRE stands for Object-Orientated Graphics Rendering Engine. It is written in C++ to facilitate the object orientated part. The fact that the engine is object orientated means that is easy to manipulate objects, like lights, meshes, cameras and other things.

OGRE supports OpenGL and Direct3D and has implemented a lot of different graphic techniques. OGRE has an integrated GUI system, called CEGUI (Crazy Eddie's GUI) [1]. This system is used for displaying GUI elements. In the game these are the health indicator, the ‘game over’ message and the mouse cursor.

The most important reason to choose for OGRE3D is the availability of resources for developers online. There are a lot of tutorials, articles and howto’s available about OGRE3D.

5.6 The Game

The map in which the game takes place is an outdoor scene. The floor consists of an infinite plane with a grass texture. On top of the floor plane there are swaying grass leaves. The swaying grass is implemented using static geometry. With static geometry it possible to place the same mesh many efficient in a very efficient way. The sky is made by a skybox. This landscape is created using an example which was deliverd with the OGRE software development kit. A screenshot from the game is visible in Figure 5.6.

In game enemies (Figure 5.7) will spawn at random times. These enemies are the targets for the player to hit with the mouse. The objective of the game is to kill as many enemies before getting killed. Sometimes the player has to make a movement with his head to actually see the enemy. When hit the enemy will perform a ‘die’ animation and after its done doing so it is removed from the scene. When an enemy is present it will walk randomly around the scene. An enemy has three states:

- **Inactive**: It is dead and waiting to be spawned.
- **Active & Rotating**: The enemy rotates to a new point before it is going to walk to there.


Fig. 5.6: A screenshot taken from the game.

Fig. 5.7: An enemy, the targets the player has to hit.

- **Active & Walking**: The enemy walks to a new point.

- **Dying**: The enemy is dying.

From time to time projectiles (Figure 5.8) will be launched at the player. The projectiles will come from random positions at a fixed distance, but they always aim for the player’s head. They will fly in one straight line so that the player can dodge them by moving to another position. By dodging the missiles the player has another way of interacting with the game. Of course it is visual nice that the perspective changes when the screen is viewed from an actual angle, but if there is no reason to move the player will soon get tired of it. The missiles force the player to actively use the head tracking system.
5.7 Collision detection

OGRE is a render engine, not a physics engine. This means that for example collision detection between objects is not implemented. Collision detection is something what is required at two points in the game. First for detecting a hit between a missile and the player. The second place is to check whether the player has hit something when shooting at a target.

For collision detection between missiles and the player a method in OGRE is used which is labeled as ‘internal’, the documentation recommends not to use this. The method `getWorldAABB` returns a Axis Aligned Bounding Box (AABB) of the missile. With this bounding box it is possible to test whether the is inside it. The bounding box is somewhat bigger than the actual missile, so a hit can occur when the head position is not in the actual mesh. This is not a problem, because the player’s position is regarded as a single point. In reality the player is bigger than one point, so when playing the game the player has to make sure he keeps enough distance from the incoming missiles.

For detecting hits on the targets an in-built ray tracer in OGRE3D is used. It shoots a ray through the projection plane at the point the user clicked and returns an iterator to iterate over all objects which intersect the ray. When an enemy is in that iterator it is a hit.

5.8 Projection Matrix

Unfortanatly OGRE does support an off-axis projection as required for a head tracking application. OGRE does allow a custom projection matrix to be defined, so the required projection is possible.

It is assumed that the imaginary window is always at position (0, 0, 0).
CHAPTER 5. A DESKTOP VIRTUAL REALITY GAME

Fig. 5.9: Both ways of rendering the perspective. The first image shows the correct approach, using shearing. The second image is rotated towards the origin of the scene (0, 0, 0)

It lies on the x axis.

The first method used was without a custom projection matrix. The camera was placed on the position of the head and then the viewing direction was set such that it always looked to the origin (0, 0, 0). This did not give a real experience. By moving from side to side indeed possible to see more of the other side. But this was caused because the view rotated, instead of sheared. In the case of rotating the projecting plane is still perpendicular to the viewing direction, which should not be the case.

To realize the correct projection the view direction is set to (0, 0, -1). The camera is still placed at the head position. As the window is at distance $z = 0$ the near plane should be set at that position. To get the effect that things come out of the window, a closer near plane is necessary. A distance of 0.05 units is used to place the near plane in the game. OGRE creates a projection matrix using the supplied viewing parameters. This projection matrix is multiplied with a shearing matrix to get the shearing effect. The resulting matrix is then used as the new projection matrix.

In Figure 5.9 both ways of creating the perspective are displayed next to each other.

5.9 Evaluation

Head tracking adds a whole new level of expierence and realism to a game. It really gives the idea of being part of the environment. The perspective changes in a realistic way when looking from a different angle to the screen. In Figure 5.10 the game is shown in action.
In the demonstration of Johnny Lee, it appeared that some of the objects floated out of the screen. He realized this by setting the near plane really close. I tried to recreate this effect, but without success. This is happens because of the landscape. The landscape should only be behind the window, for the grass cannot come through the window inside the player’s room. In the game landscape is also rendered which lies in front of the imaginary window. This creates the feeling standing between the grass, but it destroys the effect of objects getting out of the screen.

For true three dimensional viewing humans use steographical viewing. The distance to objects is estimated by comparing the vision of one eye with the other. While the game alters its perspective to the position of the head, still both eyes get the same image. For a real three dimensional experience a technique should be used which can render a different perspective for each eye.

Although not truely three dimensional the effect is still convincing especially when making small movements when objects are close to the player. The best effect is realized when sitting between the grass leaves and making small movements. This happens because humans also use another mechanism for three dimensional viewing. By combining the view taken from
different view points when moving around a person with one eye can still have three dimensional viewing. Pidgeons also do this by constantly swaying their heads.
Chapter 6

Summary / Conclusion

For this projection a head tracking system was created using the Wii Remote and glasses fitted with IR-LED’s like Johnny Lee had demonstrated. The system is very cheap. It only requires a Wii Remote, which costs only 40 euro’s, if you do not already own a Wii. Besides the Wii Remote you also need glasses with IR glasses. The glasses are available for about 5 euros and the LED’s costs are negligible, they cost only a few cents. The total costs for the complete head tracking system is about 45 to 50 euro’s, which is really cheap compared to commercially available head tracking systems.

The camera the Wii Remote is equipped with is ideal for tracking purposes. Its specifications of 100 Hz, high resolution (1024 x 768) are outstanding. It outperforms any commercial camera available in the same price class. The onboard blob detection makes it really easy to implement a tracking system of IR sources, this in contrast to using a webcam where blob detection still has to be implemented.

For this project a head tracking software library was implemented. It easy to reuse in other similar projects, which want to use head tracking capabilities. Documentation generated from the source code is available. The library is open source and can be freely modified by anyone to fit their needs. Also an UDP protocol TUIO was implemented to be able to send the head tracking data over the network and have independent applications for the head tracking system and the program itself.

To demonstrate the head tracking capabilities of the system a demonstration program was created. This is a virtual reality game which turns the screen of the player into a head tracked display.

The perspective looks correct and changes in a natural way to changes of the body of the player. This effect brings the game experience to a whole new level. The effect is not real enough to actually experience the idea that objects get out of the screen. This is because the landscape runs through
the whole scene, also in the area which should be considered in front of the screen.

For three dimensional viewing people use stereographic viewing. In the demonstration program created for this project the view is the same for both the eyes. Therefore no true three dimensional viewing is possible. For this an extension should be made to realize an independent view for both the eyes.
Chapter 7

Future Work

For research head tracking has already been used a lot, but there are almost no commercial games available which enables head tracking for the average person. Traditionally games were only controlled by a keyboard/mouse or a game controller. With the introduction of the Nintendo Wii new ways to interact with games were introduced to the public. As we have seen with the Wii, it new way of interacting has been a revolution in the game industry. The success of this revolution indicates that in the future probably even more new ways of interaction will be developed. Head tracking could be one of those, for having a virtual reality in games will give the player a lot of excitement.

Head tracking is a new application for the Wii Remote. Johnny Lee has shown it was realizable. With this projected a library was presented for developpers to create their own head tracking applications. In the future more applications for this library can be developed. Of course more games are an obvious choice, but it is also thinkable that head tracking is used as an replacement for the mouse.

There also a few improvements possible to head tracking system itself. At this moment not all six degrees of freedoms can be tracked. With another LED this will also be possible. The Wii Remote can track up to four points, so this is something which is doable.

At this moment the glasses used are just ordinary safety glasses. When replaced with shutter glasses it will also be possible to send a different image to each eye. This will make stereographic viewing possible and will give the player a real three dimensional viewing experience.
Chapter 8

Bibliography
Bibliography


