A Natural Interface for 3D Manipulation

Nurazlin Zainal Azmi

a thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand.

28 February 2014

Abstract

3D interfaces enable a user to create and manipulate 3D objects in a computer-simulated 3-dimensional world. There are many existing interfaces, but few of them exhibit the naturalness which characterizes our interaction with actual 3D objects in the real world. Developing a natural 3D interface is increasingly becoming an area of interest. Researchers are exploring several techniques to help achieve naturalness in interaction, based on the analysis of hand gestures or on tangible devices.

In this thesis, we investigate the use of a cheap and widely available commodity device to create a natural 3D user interface. The device that we use is not only cheap and available on the market, but it is also fast and allows 6DoF manipulation. Our focus is to create a one-to-one correspondence between positions in physical and virtual space. We calibrated the device to suit this purpose.

Our interface is shown to be easy to use based on an experiment in which users perform a simple assembly task. The task was performed by subjects using our new interface and also by subjects in a control condition using a traditional mouse-based 3D interface. We recorded the participants completion time in both conditions and found a clear advantage for our interface. We conclude by describing some possible future directions to make this a better interface.

Acknowledgements

I would like to express my deepest gratitude to my parents, for allowing me to pursue my PhD in University of Otago. They gave me their full support when I decided to stay here until my study is completed.

To my supervisor, Geoff Wyvill, thank you for always believing in me. He pushed me to see my own potential and helped me to realize what I am capable of achieving. Without his guidance, I would not be the person I am now.

To my co-supervisor, Alistair Knott, thank you for all your support, and especially for taking the hours of your precious time during your holiday to read my thesis.

To Janet Rountree, who was once my co-supervisor, thank you for still being there for me. Although no longer part of the committee, she has always given me support and advice whenever I have needed it.

To Niamh Fitzgerald, who always finds time to help me with English despite her busy schedule. I really appreciate everything that she has done.

To Thomas Verbeek, thank you for your work using Blender that helped reduce my workload a little bit, and for always being a friendly face in the lab.

Not to forget, to everyone in the Department of Computer Science, especially those in the Graphics Lab, past and present. You are my new family. Having everyone around especially during the hard times eased my troubles and gave me courage to get through these hurdles.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Given a 3D input device and a display screen, how can we use the data from the device and the knowledge we already have to create a natural 3D interface where we can interact effectively with a virtual 3D model?

J. Blake [12] described the changes of trend in user interface as below:

> “The world migrated from command line interface (CLI) applications to graphical user interface (GUI) applications because GUI was more capable, easier to learn, and easier to use in everday tasks... The same thing is happening again right now, except today GUI is the stale technology and natural user interface (NUI) is the more capable, easier to learn, and easier to use technology.”

The typewriter keyboard, invented more than 100 years ago, has effectively solved the problem of text input. So, we had a tool for CLI already. When the world migrated to GUI applications, the technology changed as well. Suddenly we have a mouse, pen tablets and touch pads which undoubtedly act as an effective 2D input for the users.

Now, the technology is moving towards NUI. Unfortunately we can’t say the same about the 3D input used for natural interfaces. It is really hard to define a standard framework in 3D because each researcher has his own interpretation of NUI. In this context, a natural interface is defined as *”a user interface designed to reuse existing skills for interacting directly with content”* [12]. Some researchers create a 3D interface specifically for one purpose. One example would be a flight simulator. This simulation can be used only for flight training or entertainment in this area.

There are also 3D applications that utilize 2D input as can be seen in 3D modeling software such as Blender and Maya. Although the 2D input can be used successfully with these applications, users need to undergo training in order to master the interface.

However, wouldn't it be much easier to be able to manipulate virtual 3D objects as easily as we do real objects with our hands? This is the area that we want to focus on. We are not looking to create a new piece of hardware specifically for our purpose. Rather, we are looking for a way to use the current state of the art to create a natural 3D interface. This leads to the idea of using a cheap, available device to achieve as natural an interface as possible.

Researchers in this area may decide between two design approaches. They can either create an interface that resembles the actual interaction of the real world such as the flight simulator or they can incorporate some tricks to achieve the natural interaction, which is also known as "magic" technique [14].

The task of the 3D UI always circulates among three categories which are travel, selection and manipulation [14]. The user must specify which of these three forms of interaction is taking place, before performing gestures which are interpreted by the interface. In travel, users can move around in the 3D environment by changing their viewpoint or avatar. In this work "avatar" is generally used to mean any virtual object under direct control of the user. This terminology is further explained in Section 3.2. Selection deals with choosing and picking up an object from the 3D environment. Manipulation means performing an operation on the virtual object such as moving it, changing its orientation or shaping it.

After defining the category, the next step is executing the task. As mentioned previously, researchers have to decide between the idea of maintaining the naturalness in the interaction or using some tricks that will ease the interaction.

The natural interface should represent real life action, at least that's what we believe. But there are actions which are too complicated to simulate in this context. An example is grasping. A simple action such as picking up an object with a hand requires a complex simulation. Our grasp is soft and yielding, so we need to use the physics of contact such as the force to grab the object or the friction to hold the object to guide us. This is what we call the softness of the interaction. Apart from that, we also need to consider the complexity of modelling the hand itself. Without these factors, the simulation will not be natural.

This is why a "magic" technique is introduced to avoid this type of problem. For example, in the selection task, in order for users to select an object naturally, the object has be close enough to the users. If it is further away, they have to walk towards that object until they can touch it. With a "magic" technique, they might just have to extend their arms a little bit and the simulation will bring them closer to the object.

Or they can just point in the object's direction and select the object by using a ray casting technique.

This interaction has to be implemented with 3D input. Researchers have two options. First, they can use an existing device available in the market. Secondly, they can choose to design their own 3D hardware. However, this approach requires them to take the design issue into consideration.

Since our focus is to use the existing hardware to create the interaction, we need to find hardware that meets our requirement. The physical device can be as simple as the Wiimote or the PS Wand. What matters to us is the manipulation of the interface. How fast can a user learn the new interface with only a few instructions? As argued by some researchers, natural interfaces are not always natural and can be achieved with a few tricks [14, 80]. With this understanding, we are ready to go deeper into this natural interface area.

1.1 Motivation

We have explained the need to create NUIs but would they be a better interface? As we know, some of the real world interfaces are awkward. If we look at a mechanical digger, its interaction is not direct. The interaction is controlled by using a combination of joysticks to perform a task. Sports such as volleyball take time to master. The hand is a totally "natural" interface, but learning to control its movement to hit or serve or set the ball requires a lot of effort and practice. Even passing the first ball to a setter demands frequent training. It would not be wise to create such a complicated interface for a 3D virtual world but it is worth investing the time to create a 3D interface that can be used naturally.

To find out if a 3D NUI is in fact better than a traditional interface, we need to:

- set up an environment where we can test a natural interface; and
- create a suitable example test that enables us to make a start.

What could be a good task to be performed in a 3D NUI? Many tasks in the real world require assembling objects. Even our every day actions such as putting objects in a shopping bag, changing a light bulb, and replacing a camera lens require the actions to be done accurately.

Simulating the assembly of objects is not new. There are a lot of 3D modeling programs that can be used to simulate this type of action. Examples are Maya, Auto-

CAD, 3ds Max and Blender. These do the job but the manipulation of the 3D objects is not made easy with these programs. The action takes much longer than it would in the real world.

For an instance, if someone wants to attach a tyre to a car, he just has to bring the tyre to the car and attach it to the right wheel. If the same action is to be done with a 3D modeling program, the task won't be as simple as that.

In the modeling software, the task can only be completed by using several modes and views. Assuming we are looking at the front of a car. The tyre will be on the side of it so we have to change to the side view. To move the tyre will require another different mode. We can't move it freely in 3D space but it has to be done in a constrained movement which usually means we can only move it in one axis direction at a time.

To attach the tyre correctly, we have to make sure that the tyre is properly aligned with the wheel's position. We might have to lift the body of the car a little bit and rotate to align them properly. Rotation, like translation, is usually restricted, which means the car can only be rotated about one axis at a time.

This restricted movement doesn't even have naturalness in its interaction and it doesn't make the task a lot easier either. This is the greatest weakness of the interaction metaphor used by these programs. It should be possible to perform this task, in simulation, using the same actions as we do in the real world. That is the motivation for this project.

1.2 Objective of the Thesis

Based on the importance of the assembly task in the real world and the need to simulate the same action in a virtual world, we define our goal as:

> "To create an application where users are able to manipulate a 3D object as if they were doing it in the real world, using readily available hardware."

The objective is to find out if users can perform the task easily and accurately without any problem while manipulating the 3D object in 6DoF[1].In doing so, we need to make sure that the simulated world is implemented with the correct physics. This is important because the real world has physics and we can use the physics to achieve the softness of the interaction.

[1]DoF means "degrees of freedom".

With the goal in our mind, we need to properly define what a NUI is. Based on J. Blake's definition of NUI [12] as stated on page 1, this definition covers three important factors about NUI:

- NUIs are designed

  As we have already seen, NUI requires a natural interaction between the user and the hardware. To achieve this naturalness, we need to plan carefully and design its interaction. We have to make sure the design is suitable to be used with the application.

- NUIs reuse existing skills

  As stated in Section 1.2, a natural interaction can be achieved using the experience we already have. If we are creating an interface where it is too complicated and requires a long time to understand, then we have already violated the definition.

- NUIs have direct interaction with content

  This third point carries the meaning as it is. The primary interaction method with the application is, and must be through direct manipulation.

Since NUI requires the reusability of existing skills, we feel it is important for users to be able to interpret the nature of an interface the moment they look at it. For example, given a glass, one should straight away realize the main purpose of the glass is to contain water and drink from it. Using the same concept, we want users to have the same feeling when they look at our interface. We want them to be able to figure out what the interface does and with only a few instructions, they should be able to perform a 3D manipulation using the interface.

Since we focus on a task in 3D manipulation, we can also be more precise about which skills we expect users to have. A user needs to learn the relationship between the display and the 3D world portrayed, but if we are successful, the visuomotor skills for the manipulation itself should already be there from everyday experience of moving objects. So we use a modified version of Blake's definition and define a natural user interface as:

> "*An interface that relies almost exlcusively on visuomotor skills already possessed by users.*"

1.3 Focus of the Project

Our work has five important characteristics.

- We used cheap hardware for the interface.

  Popular hardware such as the Nintendo Wii, the PlayStation Eye and the Microsoft Kinect are used by the majority of gamers. This fact can be an advantage for creating a natural interface since a lot of people are used to this hardware already.

  These hardware platforms also provide their own libraries that allow the device to be controlled by a computer, hence making the programming of the interaction much simpler and faster. Some companies provide support for their libraries which is an added advantage.

- The interaction is 6DoF.

  Our actions in the real world are only limited by the laws of physics and the shape and strength of our bodies. We are free to move and turn in any direction we want; we are not limited to making movements in a single selected plane. For a natural interaction to occur in a simulated world, these real life actions must be simulated as closely as possible. This is where the 6DoF interaction comes in.

  Degrees of freedom (DoF) define the ways in which a rigid body can move inside a space. In total, there are only 6DoF; these can be divided into two components: translaton and orientation. Inside a space, a body is free to translate in 3DoF: forward/back, up/down and left/right. Another 3DoF comes from rotation in space, in the dimensions of pitch, yaw and roll. If an object can move freely in all three types of translation and rotation, the movement is said to be in 6DoF.

- It focused on a 3D manipulation task.

  There are a lot of 3D manipulation tasks that can be simulated in a virtual world. The task that we are looking for is an assembly task that involves translation and rotation. We choose this type of task because assembly is the most common activity done in the real life. As a matter of fact, almost everything we do deals with putting things together!

- The interface was tested by experiment.

An interface is created to be used by other people. If it is not tested by the intended users, the main goal of creating the interface might not be met. Testing is what determines the success or failure of a system.

- We compared our natural interface with a mouse interface on a similar task.

  It is important to test not just the interface but to also compare it with a related application. In our case, we chose to test the same task on two different styles: the NUI vs the standard interface. We want to see how our interface performs against the current standard modeling software doing exactly the same task. The current modeling software such as Maya, 3ds Max and Blender uses a mouse and a keyboard as an input.

  To modify the curent standard modeling software to use the same input or interaction as ours would be unfair. Therefore, we decided to observe users' performances when completing the task with our interface as well as with the current standard modeling interface.

1.4 Limit of Scope

Naturalness is defined by a combination of factors. In an ordinary assembly task, we have a sense of touching the object. We experience resistance when we try to put a nut in a bolt if it is not of the right size. When our groceries fall out of the shopping bag, we hear a sound when they touch the ground.

To simulate all these effects will be impossible for a single PhD project. These are our limitations. We do not include sound nor haptic feedback in this interface. We are only looking at a simple assembly task where our focus is to achieve naturalism through a one-to-one correspondence. The manipulation of a 3D object in the interface is controlled only through a "magnetic sticky grip".

The observation of the virtual world is made only with an ordinary screen. We do not incorporate stereo vision to produce a 3D effect. There is also no motion parallex as in a virtual reality system.

1.5 Contributions

The contributions of our research can be summarized as follows:

1. We have created a test bed for natural interaction experiments that:

   (a) uses a physics engine and a 3D modeling package to model assembly tasks; and

   (b) provides a one to one 6DoF input with a cheap game controller.

2. We have set up a system to compare performance of a simple assembly task with the same model using a conventional interface.

3. We have performed an experiment demonstrating the ease of assembly in one particular task, using volunteer human subjects.

4. We have established a simple way to calibrate a game controller to make it useful for this purpose.

1.6 Thesis overview

This thesis consists of eight chapters. We discuss the history of interfaces, what motivates researchers to pursue this field and work that uses the direct manipulation technique in Chapter 2.

With the knowledge of the work done in the natural interface area, we present our conceptual design in Chapter 3. In this chapter, we try to understand the problem and analyze the best possible way to achieve softness in interaction. Chapter 4 touches on the detail of the implementation. This includes the choice of software and hardware to use. We dedicate the whole of Chapter 5 to explaining our calibration process, since this process is essential for obtaining a natural interface.

The next two chapters present the experiment and its results. We talk about the experiment carried out in Chapter 6 and discuss the result in Chapter 7. We conclude in Chapter 8.

Chapter 2

Background: User Interfaces and 3D Interfaces

Our objective is to create a 3D interface that incorporates natural interaction as in the real world. We have developed an application that fulfilled this requirement. In this chapter, we briefly discuss the history of user interfaces and techniques used by researchers to develop the so-called 3D natural interface that are related to our project.

2.1 The History of User Interfaces

Computers have now become an essential element in our daily routine. It is not just a matter of what the computer can do, but also how people can interact with it. That's the importance of the user interface - "a part of a computer system through which human user and computer communicate" [110].

Since the creation of user interfaces, the way of interacting with the computer has evolved tremendously. From commands to graphics, and now to gestures, there are still a lot of elements in this field that can be developed further.

2.1.1 Batch Interface (1945-1968)

In batch computing, users describe to the system the requirements of their tasks, using "job control languages". The interface of batch computing uses a job card as an input.

To submit a job to the batch machine, a job step which consists of statements that control the execution of a program and request of resources is described on the job card. The job step also specifies the input and/or output of the program.

A normal turnaround time for the batch machine to process a single job was about an entire day. All the jobs were run as background work, which meant the jobs did not require user interaction.

2.1.2 Command Line Interface (1969 - present)

The job control languages were later renamed as command languages in interactive systems. This requires an interaction between a user and a computer program, or between two computer programs.

To give an instruction to the computer to perform a specific task, the user needs to pass a line of command in the form of text to an interpreter. This line is interpreted accordingly and the system returns the response from the interpreter. Another command may be passed when prompted. This process is repeated until the task is done.

2.1.3 Graphical User Interface (1981 - present)

As the computer evolves into a more powerful and advanced machine, the user interface also follows the same shift. Instead of typing a line of text, a shortcut such as icons and menus is used to perform a task.

This graphical user interface or GUI makes it easier for the users to interact with the computer without having to deal with the complexity of command languages. It uses a pointing device such as mouse that enables users to select commands on a display screen.

I. E. Sutherland [103] wrote a computer program called Sketchpad for his PhD thesis. Considered to be an ancestor to computer-aided drafting programs, Sketchpad become one of the most influential computer programs written by an individual. Its development contributed to studies in Computer Graphics and Computer Interaction.

Later, in the same year, D. Engelbart [28] created the first mouse prototype that can be used with windows. Like I. E. Sutherland [103], he also opened a way for computer interaction to evolve.

The first GUI was developed to be used in the Alto personal computer in 1973 by Xerox PARC [26]. The GUI consists of windows, icons and menus, and supports basic commands such as opening, deleting and moving files. In 1981, Xerox introduced its pioneering product, Star, as a commercial GUI.

Influenced by the Alto, the Macintosh was released in 1984. This computer used a desktop metaphor and became commercially successful. The GUI continues to develop, and in recent systems includes 3D effects in the desktop environment.

2.1.4 Natural User Interface (2006 - present)

NUIs are a different class of interface from GUIs and CLIs because they are defined on orthogonal dimensions. NUIs are defined as being interfaces which exploit existing skills of the user (for instance existing visuomotor skills). This definition makes no reference at all to the devices used in the interface. GUIs and CLIs, on the other hand, are defined as interfaces which use certain classes of interface device. For instance, a GUI by definition uses a visual display and a pointing device. A CLI by definition uses a textual input device. It is **possible** that a GUI is also a NUI, if control of the pointing device also happens to exploit an existing user skill. But it is also possible to imagine a GUI which is **not** a NUI. Equally, it is possible to imagine a NUI which is not a GUI (for instance, a speech-based interface).

In Section 1.2, we gave our definition of NUI. We looked at J. Blake's interpretation [12]. He also touched on the meaning of the word 'natural' to give us a better understanding of NUI. He quoted the definition of that term from Bill Buxton's interview with Larry Larsen [66] as an action that "*exploits skills that we have acquired through a lifetime of living in the world.*"

It is worth looking at what others have said about the concept of naturalism as well. K. Hinckley and Daniel Wigdor [47] wrote:

> "A reasonable working definition for *natura* is that the experience of using a system that matches expectations such that it is always clear to the user how to proceed, and that few steps (with a minimum of physical and cognitive effort) are required to complete common tasks. It is a common mistake to attribute the naturalness of a product to the underlying input technology. A touch-screen, or any other input method for that matter, is not inherently natural."

According to B. Alun-Jones [5] in his report:

> "One main way in which HCI systems need to change in is their intuitiveness. Computers are not easy to use, you have to learn many skills of high cognitive load in order to be able to use them effectively. By making interaction more natural and tailoring it to the way in which we, as humans, behave can reduce the mental challenge of learning these new systems."

R. Raisamo [86] described the term 'natural interface' as follows:

> "The term *natural user interface* is not an exact expression, but usually means an interface that is easy to use and seamless as possible. The feeling of using an interface can be faded out by not attaching any interaction devices to the user and by designing the dialogue in a way that is really natural and understandable for the user."

Based on these definitions, it seems as though most researchers agreed that for an interface to be natural, it has to be easy to use without taking too much effort to learn it, and the interaction must be closely reflect a common human behavior.

2.2 User Interface Studies

3D interaction is not new. In 1983, B. Shneiderman [95] published a work on direct manipulation where he described it as:

> "...visibility of the object of interest; rapid, reversible, incremental actions; and replacement of complex command language syntax by direct manipulation of the object of interest..."

With the trend moving towards NUI, the amount of research done in the 3D user interface area has increased a lot. Researchers share one common objective, which is to make the communication between the user and the computer easier. As mentioned in Section 1.3 in Chapter 1, we focused our work on five important characteristics.

To our knowledge, none of the related work we have found does all of these things but there is a lot of interest in the same ideas. In the ACM Digital Library alone, we found 7457 papers related to 3D natural interfaces. From this pool of papers, a lot can be excluded especially those whose work was based on natural language, audio interface, 3D sketching and animation.

2.2.1 Natural Input for 3D Manipulation

In this first part of our survey we examine work that uses cheap commodity devices to create 3D interfaces. In particular, a lot of work has used game controllers and mobiles because these are so readily available.

2.2.1.1 Using the Microsoft Kinect

P. Song et al. [99] used a low cost sensing device, the Microsoft Kinect, to perform a 3D manipulation task. They implemented a handle bar metaphor that maps the user's hand gestures to the corresponding virtual object manipulation operations.

The authors argued that the current mid-air interaction methods for manipulating a 3D object normally require contextual and mode switching to perform operations such as translation, rotation and scaling. Hence, their aim was to develop a controller-free environment that supports natural and intuitive mid-air interactive gestures without using any cumbersome handheld peripheral.

The proposed handle bar metaphor mimics common situation of handling objects that are skewered with a bimanual handle bar. This interface uses hand gestures - POINT, OPEN and CLOSE - to control the interaction.

The 3D manipulation was done by first selecting the object in the Browse and Select mode. A hand avatar was used to represent the movement within a 2D plane that follows the OPEN palm gesture. Once the object was selected, a virtual handle bar would be pierced through it. The translation, rotation and scaling was performed by manipulating the handle bar combining the hand gestures.

The interface was evaluated in three different tasks. The first task was for rotation and translation, the second task was for a constrained rotation and the third task was for multiple objects' manipulation and alignment. The constrained rotation was introduced because the handle bar metaphor has one weakness - a rotation around x-axis is not possible because "the wrist-based rotation of the two CLOSE (hand gesture) fists does not change the position of their (3D virtual objects) centroids and thus gives no angular rotation cues".

M. Raj et al. [87] also used Microsoft Kinect as a cheaper way to support a 3D manipulation task on a desktop display. Their purpose was to evaluate a gesture-based method of interaction that is more suitable for a 3D operation.

For this purpose, they implemented two visual display controls - a self-avatar hand with an arm and a sphere as shown in Figure 2.1. They wanted to see if the avatar would facilitate the task of 3D object manipulation.

In the experiment conducted, the participants were asked to manipulate the object on the right side of the display screen to match with the orientation of the object on the left side of the same screen. Each participant was equipped with a sensor - Intersense InertiaCube3, and a mouse on their right hand and was given a starting position.

Two modes were used to manipulate the object - swipe and twist. The swipe mode

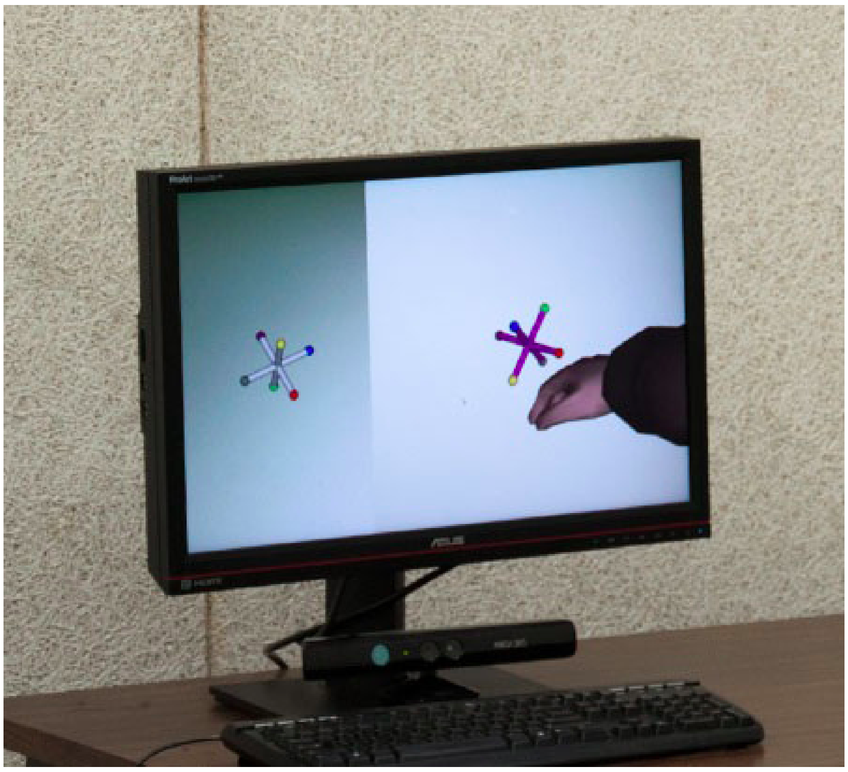


Figure 2.1: A self-avatar hand with an arm and sphere, taken from M. Raj et al. [2012]

is used to drag the object in the direction of the hand whereas the twist mode means the changes of the orientation of the object about the wrist axis. If the orientation of the manipulated object was within 15 degrees of the orientation of the target object, the participants would be prompted with a "match detected" message on the display screen. They were only given 90 seconds to complete the task.

As mentioned earlier, the authors' goal was to find out which visual display could facilitate the 3D manipulation better. From the conducted experiment, there was no overall difference in performance found between these two visual displays. The participants' gender and video game experience did not affect the performance in the self-avatar display condition. However, these two factors did influence the manipulation task in the sphere display condition. Since these individual differences only occured in the sphere display condition, the authors suggested that the use of self-avatar would be beneficial to a larger population of users.

Another project that took advantage of the popularity of Microsoft Kinect was done by O. Hilliges et al. [45]. A system called HoloDesk which combined an optical see through display and the Kinect camera was developed to interact directly with the 3D graphics as shown in Figure 2.2.

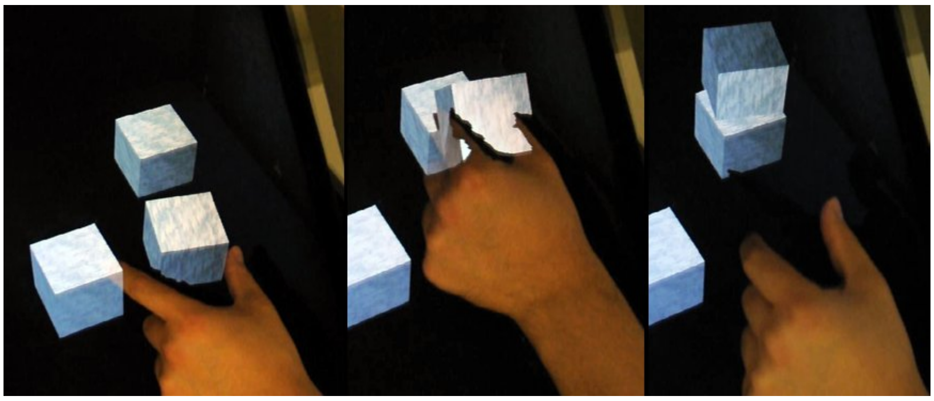


Figure 2.2: HoloDesk, taken from O. Hilliges et al. [2012]

Their aim was to allow the users' hands to be literally inside the display so that they can interact with the virtual objects without the need to wear any specialized hardware or input devices. This virtual world needs to be spatially aligned with the real world for the interaction to take place. To make the interaction more natural, the authors used a hand grasping method to manipulate the object.

The authors claim that grasping is arguably the most common mode of manipulating objects. They also stated that to model a hand grasping technique, the collision and forces exerted on the virtual objects need to be modelled accurately.

Also using the Kinect, R. S. Yang et al. [114] proposed a way to control a 3D modeling application based on a low cost system. The authors also applied markerless tracking to create a more natural interaction.

They developed a prototype system that uses 3D hand tracking and posture recognition to perform 3D modeling. They were looking to explore the usability of using bimanual control to perform the interaction. They wanted to find out if the hand posture and motion are more natural than using a mouse, and also to identify if the combination of posture and hand movements are a good way to use both hands for manipulating the 3D object.

In this bimanual interaction style, the left hand is used to command selection via hand postures. The right hand is used to control the movements in the virtual 3D space by using the real 3D world coordinates of the hand.

The interface was evaluated where the task was to move an object in the 3D space to a target location. The same task was also studied using a mouse interface. The initial starting position of the 3D object was set at random, but the final position was always the same. As expected, from the result, the participants preferred to use the hand tracking system to perform the operation.

2.2.1.2 Wiimote and Kinect

R. Francese et al. [32] developed natural interfaces using two popular gaming devices: Nintendo Wii and Microsoft Kinect. The authors exploited the popularity of these devices to create an interaction that can be expanded into 3D.

Their goal was to replace the traditional point and click interaction of the Bing Maps classic PC navigation with gesture-based navigation. This idea was demonstrated in two applications known as Wing and King.

In Wing, the Wiimote is used together with the Nunchuk to control the navigation. The interaction on the Wiimote was implemented based on the motorcycle metaphor. The device was used to control forward/backward movements as well as turning. The Nunchuk on the other hand was implemented with the aeroplane cloche (joystick) metaphor where it was used to control the altitude during the navigation.

In King, a paper aeroplane was designed as an avatar in the 3D world representing the gesture performed by the user. The application uses a bird/aeroplane metaphor. The Kinect controller captured the flying gesture performed by the user and mapped the movement to the avatar. They proposed the bird metaphor is a novel interaction that hasn't been used in gaming.

Student volunteers without particular competences in 3D virtual world, games and NUIs were used as subjects for testing. Each one of the them was instructed individually on how to use both interfaces and then was asked to complete the navigation tasks on two geographical paths of well known Italian cities. Their comments on the interfaces were taken into account and their behaviors were observed. After the test, they were required to complete questionaires.

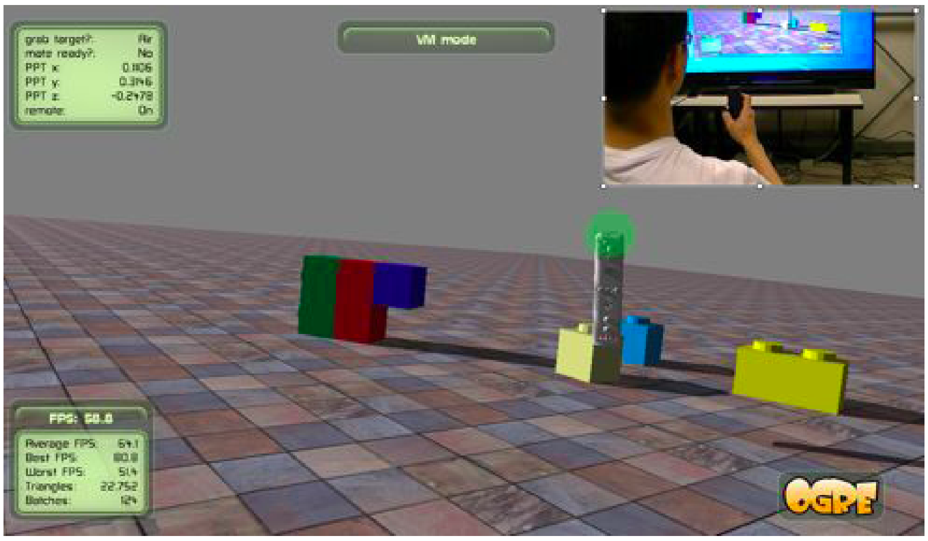
2.2.1.3 Using the Wiimote

M. Chen et al. [21] built a 3D controller combining the Wii and several cameras. The aim was to develop a universal motion control from the current and cheap hardware that can perform general tasks in both 2D and 3D by simply changing its mode of operation.

The authors stated that "*the WIMP GUI may still be irreplaceable but 3D UIs will be superior when the interaction is taking place in a 3D spatial context*". Because of this, they suggested that the universal user interface should be something that can combine both 2D and 3D interaction.

They implemented this interface and demonstrated how it looked for a couple of tasks including 3D manipulation of virtual Lego Bricks as shown in Figure 2.3. In their demo application for 3D manipulation and 3D navigation, a lot of cues were used to tell the users what interaction they were performing. A red overlay is used if an object is graspable or objects are engaged but not correctly aligned. If the objects are correctly aligned, a green overlay will be shown and the user can press a button that will trigger the 'assemble' command. For 6DoF browsing in the application, the user needs to point the controller upward until a semi-transparent green appears which indicates the desired operation has been triggered.

A universal motion controller is an interesting idea but it cannot be said to be natural. It is a bit like having common steering system for cars, bicycles and boats. Furthermore, there is no report of user testing so it is not possible to see whether the universal interface is in any sense better.


```
grab target?:   Nr
mate ready?:    No
PPT x:      0.1106
PPT y:      0.3146
PPT z:     -0.2478
remote:         On
```

```
FPS: 60.0
Average FPS:  64.1
Best FPS:     60.0
Worst FPS:    51.4
Triangles:  22,752
Batches:       124
```

Figure 2.3: Virtual Lego Bricks, taken from M. Chen et al. [2011]

M. Resl et al. [89] compared three Wiimote-based 3D interaction techniques for basic spatial object manipulation. Instead of using gesture recognition, they mapped the gesture movement to aid a 3D manipulation.

The first technique used only a single Wiimote. The second and third techniques used two Wiimotes but with different styles of interaction. In the first technique, the interaction is almost straightforward. The user points at an object, which is a cube in this application, and presses the A-button to pin the cube for rotation. To grab the cube for movement, the B-button is used instead.

The second technique is known as Dual Wiimote - Spinning Top. This technique uses the same concept as the first technique for pinning and grabbing. To rotate the object, the users need to move both of the Wiimotes in opposite directions as if they were starting the rotation of a spinning top with their fingers. The third technique, Dual Wiimote - Distributed Control, used different Wiimotes for a different control. The Wiimote on the right hand is used for movement while the left hand is used to control the rotation of the cube. Unfortunately, rotation around the z-axis is not possible except for the third technique where the rotation is performed by turning the left-hand wrist.

Testing was done to compare the performance of the three techniques. The task was to move and rotate the cubes in the 3D space. Based on the result, the first technique was the easiest to use and the second was the worst. The third, while not being the easiest, was actually preferred by the users.

S. Han et al. [40] took a different approach. They understood the role played by the commercial game console in the user interface area especially the Wiimote. Since the Wiimote provides only rough 3D position, roll and pitch, they proposed a system that provides a more flexible and freeform gestures compared to the Wiimote.

In this system, the authors developed a two-handed spatial 3D interaction technique using remote control devices with a hybrid 3D tracking system that can intuitively manipulate a 3D object in the virtual environment. The remote device consists of an ultrasonic receiver, inertial sensors and a switch.

The object manipulation can be performed as a one-handed or two-handed task. Selection and translation can be considered as one-handed tasks. Rotation and scaling require a two-handed interaction. They conducted a test to determine the ease of use of the proposed system where the success rate of the 3D manipulation tasks were evaluated.

L. Chittaro and R. Sioni [23] implemented a laser pointer-style (LPS) interaction based on the Wiimote to study the physiological effects of the LPS interaction compared to a mouse and keyboard setup. The task was on object arrangement for building a 3D virtual environment. This involved a 3D navigation for finding and going to the place of a given object, selecting an object and manipulating it by changing the object's position, orientation and size.

To make the interaction as simple as possible, the authors only allowed for one type of manipulation to take place at a time. The switching from one manipulation type to another is controlled by pressing key 0 on a keyboard or using a nunchuck for the LPS interaction. Each manipulation type is represented by a different cursor icon on the screen.

When the object is dragged vertically, it will be translated into a far or near movement, whilst a horizontal drag will move the object to the left or to the right. Rotation changes the orientation of the object along its vertical axis. A clockwise rotation is determined if the cursor is dragged to the left, and a counter clockwise rotation is determined by the opposite direction. To scale the object, users will just need to drag the cursor up or down.

Before the experiment, the participants listened to a short briefing about the task, about the use of physiological sensors and the use of the two webcams during the test. The task needed to be performed in four different conditions. Before performing each task, they were given unlimited time to practise the object arrangement task inside the virtual environment to familiarize themselves with the controls and the task. They were also allowed to ask questions during this trial session.

During the experiment, the participants were required to arrange a blue “K” object and a green “Z” object that has yet to be selected, to match the position, orientation and scale of a red, semitransparent copy of the object itself. To find out if the target and the object match with each other, error tolerance thresholds were used. These thresholds values were determined during a conducted pilot test to obtain a level of complexity that is acceptable for users.

At the end of the experiment, the physiological sensors were removed and the participants were asked to fill in a questionnaire, ranking the four conditions of the task performed. From the result, they found out that the LPS interaction caused more muscle exertion than mouse and keyboard.

2.2.1.4 Using Smartphone

D. Lee et al. [67] proposed the use of a smartphone to perform a 3D interaction. The authors used the same reasoning for choosing a mobile device as a 3D input - it is cheap and owned by many people.

The mobile is used as a tracker device to manipulate the 3D object. Unlike wireless controllers that use Bluetooth to connect to a system, this application uses Wi-Fi to send data to the computer.

The manipulation task was done based on Continuous Commands or Event Commands. Continuous Commands follow the movement of the mobile and are used for 3D hand placement and object manipulation. Event Commands need to be triggered by tapping on the mobile screen before the users can perform grasp and release and before entering the scaling and rotation mode.

This type of interaction has been used a lot in games especially for the mouse-based games. Gamers would have to enter a different mode before they can perform a specific action. Judging from informal observation, the proposed interaction would be easily adopted by many people.

2.2.1.5 Summary

From the first part of this review, it could be said that most of the work in natural interface has required the use of 'mode' in order to perform some types of manipulation. Although this could mean a hindrance and make the interaction less natural, sometimes the design of the interaction technique demanded the use of 'mode' in order to function as naturally as possible. This shows that it is not yet possible to develop a fully natural interface without compromising the naturalness with a non-natural interaction.

2.2.2 Innovation in Natural Interfaces

In the second part of our review, we discuss work that explores other ways to achieve naturalness in 3D manipulation especially through the use of tangible interfaces. There is also work using direct-touch tabletops.

2.2.2.1 Gesture-based Interaction

As did other researchers who used Kinect to track hand gestures, G. Bartoli et al. [8] opted to use the same device to develop a prototype system to simulate virtual first aid scenarios. Their aim was similar to the flight simulator application - to provide training for Emergency Medical Technicians to operate in harmful and critical situations. In this system, two types of interactions are used. The combination of mouse and keyboard is used to select actions the users want to perform from the displayed 3D menus. Gestures are used to perform the selected actions.

S. A. Iacolina et al. [65] developed a system to allow casual users to inspect 3D objects through hand manipulation as in the real world. The camera tracks the hands open/close gestures which resemble the act of grasping a real object. The authors compared the different manipulations performed based on free-hand interaction and multitouch. The authors claimed that the ability to directly manipulate the 3D models greatly simplified the action.

L. Zhang et al. [119] also used a camera for 3D gesture interaction but instead of using a game controller input, they used a handheld camera. They developed an application known as UCam that can directly map the hand's movement in 6DoF space to a virtual object. However, the gestures tracked by UCam are not purely 6DoF. The camera can differentiate up to 4DoF movements, and the other two degrees of freedom are achieved through the use of a control button. They compared their 3D manipulation task with a mouse input. From the result, the handheld camera seemed to be more unstable and less accurate than the mouse but the participants found it to be more flexible and more convenient to use than the traditional input device.

S. Kratz et al. [64] opted to hold a mobile device as well as a way to detect mid-air gestures that can be used for manipulating a 3D virtual object. However, rather than using a computer to display the virtual world, they used the mobile device itself as a display. The application, known as PalmSpace, creates a 3D space around the device where the manipulation of the 3D virtual object via hand gestures takes place. Two interaction techniques are used: the BackSpace and the SideSpace. These are two-handed interactions where one hand is used to hold the mobile device while the other one is used for interaction. These techniques were compared with a virtual trackball. From the result, PalmSpace performed better than the virtual trackball. In terms of qualitative ratings in public situations, based on a questionnaire evaluation, the virtual trackball outperformed both PalmSpace methods as all participants would use the virtual trackball in public spaces. The authors believed the reason was because the touch-based screen interaction is widely known compared to the PalmSpace interface which was still new.

D. Kim et al. [60] used a camera to recover a full 3D pose of the user's hand in a system known as Digits, a wrist-worn sensor. The camera is attached to the wrist so that it can image a larger part of the user's bare hand. The user can perform continuous or discrete hand gestures that support spatial navigation, pointing or selection in 3D. The authors only performed an initial test to find out how easily it can be used in hand tracking. The result showed that the tracking performance is close to existing data gloves.

Research in direct manipulation is also conducted in the Computer Aided Design (CAD) area. R. Wang et al. [106] developed a prototype for a bimanual tracking system using the PS3 Eye camera that provides a 6DoF control for a 3D assembly task as shown in Figure 2.4. The system tracks 6DoF for each hand and a pinching gesture for selection. The gestures can be used to manipulate the camera perspective and the objects in the scene. Visual cues are used to differentiate each assembly action. This prototype generates the exact positioning constraints used in CAD for mechanical engineering. Since it tracks the hand without gloves or markers, the user can easily use this system with a mouse and a keyboard.

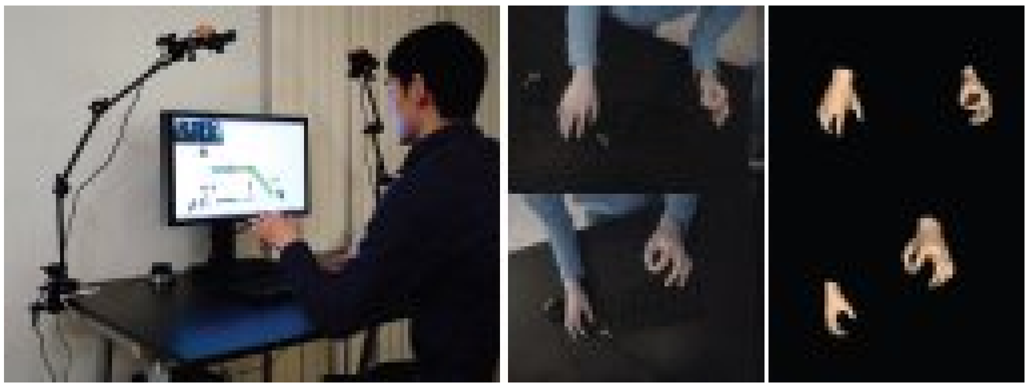


Figure 2.4: 6D Hands, taken from R. Wang et al. [2011]

Medical imaging is another area that can benefit from this 3D manipulation technique. L. Gallo et al. [34] used the Wiimote as a 3D input to interact with volumetric medical data in a semi-immersive virtual environment. This method is specifically designed to select and manipulate 3D medical data. The interaction has two states: pointing and manipulation. The user has to push a control button to switch in between the two modes.

2.2.2.2 Hand Gesture for an Open Source Framework

F. Pedersoli et al. [82] chose to develop an open source framework for Kinect known as the XKin to enable a more natural and intuitive hand-gesture interactions. Its APIs allow the user to build up intuitive and personalized applications based on hand posture and gesture recognition. The hand posture is a static pose of a hand that communicates a particular message while the gesture is the movement of hands. Their aim was to encourage researchers to contribute to the open-source community.

J. G. Sheridan [94] also went for a similar goal. She developed a prototype open source tool for two-handed multi-dimensional and 6DoF motion tracking and visualization. The aim was for other researchers to use her low-cost and "low-fi" system to capture a very fast movement of two hands in 6DoF using the Wiimote as a baseline platform. Without using the Wii development kit or the Wii tracking library, she used a camera to track the Wiimote with an LED attached to the end of it. Her next plan is to combine her current methods with the Kinect technology.

2.2.2.3 Tangible Input

There is also work done using tangible input which is not based on cheap comodity devices. D. Bradley and G. Roth [15] created a tangible user interface (TUI) system that tracks a 6DoF pose of a sphere in a real-time video stream and applies the pose to a virtual object. Red and green dots on the sphere are used to determine the orientation. The positioning is calculated by projecting the sphere onto an image plane and finding its position in the real world coordinates. The authors claimed that the sphere with 6DoF position and orientation was efficiently detected from the video stream.

D. Burnett et al. [19] developed a TUI that can be used for games running on commercial touch screen tablets. The authors aims were to ascertain the benefits of using physical game pieces on this particular device as well as to generate a framework where the contraints imposed by the operating system of the device can be evaluated by other game piece designers. They implemented a game called Air Hockey that can be played by using physical or virtual mallets as input and an iPad as the table with a virtual puck. Based on an experimental test, they could see the difference in interaction comparing their visual system with physically playing the game.

J. Lee et al. [69] used a different method to physically manipulate a 3D object. They implemented an input device known as Beyond that actually collapses in the physical world when pressed against a screen and then appears in the virtual world. With this method, the user can directly select, draw and sculpt in the virtual world without having to wear special glasses or wearables.

Another specialized TUI input, the ActiveCube, was developed by R. Watanabe et al., as shown in Figure 2.5 [107]. The user can interact with a 3D environment by constructing physical cubes equipped with input/output devices. Each cube has a unique function and its functionality can be dynamically changed depending on the connected positions/orientations or the assembled object shape. These cubes need to be connected to a base cube to allow a communication between the cubes and a host PC.

Augmented Reality (AR) is another field that focuses on a realistic interaction with the virtual world. A. Henrysson et al. [43] developed a mobile phone application in which the users can manipulate the virtual objects shown as part of an AR environment.

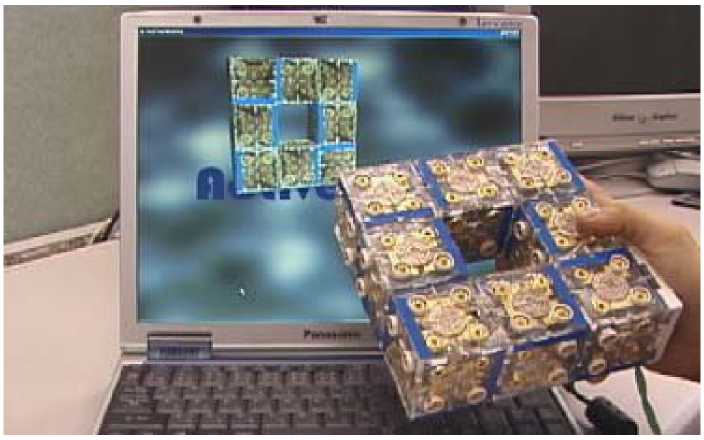


Figure 2.5: ActiveCube, taken from R. Watanabe et al.

The authors implemented several interactions for positioning and rotation. The mobile device itself can be tangible where the selected virtual object is fixed to the movement of the phone or can be manipulated through keypad/joypad or arcball. From the experiment, the authors found out that the tangible approach may be fast but is less precise and unsuitable if the interaction requires an accurate object manipulation.

There was also a study conducted by A. N. Antle et al. [6] that compares the effectiveness of TUI input against a mouse input. They used children as subjects to perform a test on solving a virtual puzzle task. Three types of actions were observed during the test: direct placement, indirect placement and physical exploration of the problem space. From the observation, the authors found that the direct manipulation based on TUI input solved the task faster and was easier to use. It also supported more exploration of the problem space than the mouse input.

2.2.2.4 Mouse as 3D Input

Although a mouse provides 2D input, it was used as input for 3D manipulation more than 10 years ago as demonstrated by G. Smith et al. [97]. The authors used constraints to restrict the object's motion in the virtual scene. Three types of constraints were used for comparison. Unconstrained (UC) mode is similar to CAD programs where the user can place an object anywhere in 3D space, with any orientation. Partially Constrained (PC) mode uses a more general constraint environment where objects can only lie on a horizontal or a vertical surface. Fully Constrained (FC) mode defines a set of contraints associated with each object. The result showed that the performance in UC is slower than FC and PC.

In the early 90s, there was also an attempt to create a 3D mouse by D. Venolia [105] that can directly manipulate 3D objects. This experimental 3D interface used a 3D cursor designed as a cone shape to control the objects. To select an object, the tip of the cursor is placed inside the selected object. The object can then be moved and rotated by using a technique based on a "tail-dragging" metaphor. To align the object, a "snap-to" technique is used.

B. Froehlich et al. [33] developed two 6DoF input methods called the GlobeFish and the GlobeMouse. The GlobeFish is a custom 3DoF trackball suspended in an elastically connected frame that can be moved slightly in all spatial directions by using force. The GlobeMouse on the other hand is a hybrid of the GlobeFish and a SpaceMouse where the trackball is placed on top of the SpaceMouse. These two prototype devices were compared with the SpaceMouse where they performed better in a docking task that required they user to rotate and translate a 3D cursor onto a 3D target.

2.2.2.5 Tabletop Display

S. P. Smith et al. [98] developed an application that can remotely rotate a 3D object on multi-touch tabletops. The authors explored two techniques using relative and and absolute mappings. In the relative mapping, a virtual touchpad is used to associate with a target object. In absolute mapping, three different metaphors are used. A "voodoo doll" metaphor copies the target object with the same properties and is rendered larger, a "telescope" metaphor enlarges remote objects and a "virtual-flying camera" metaphor provides user with an additional viewport into the 3D scene. From the test result, the relative mapping was faster than the absolute mapping. From the three metaphors used in the absolute mapping, the voodoo doll was found to be the easiest whilst the flying camera was considered too complex to use.

Another 3D application done on a tabletop surface was carried out by D. Mendes et al. [76]. The application known as LTouchIt is used to create 3D models for an interactive LEGO application and supports bimanual multi-touch input as ahown in Figure 2.6. The interaction uses a "pick" metaphor to grab and then move the object without releasing the gesture. A combination of touch gestures are used to perform translation and rotation. LTouchIt was compared with two LEGO applications that use "windows, icons, menus, pointer" (WIMP) paradigm and was found to be in tune with its competitors but it gave the advantage of having a hands-on experience.

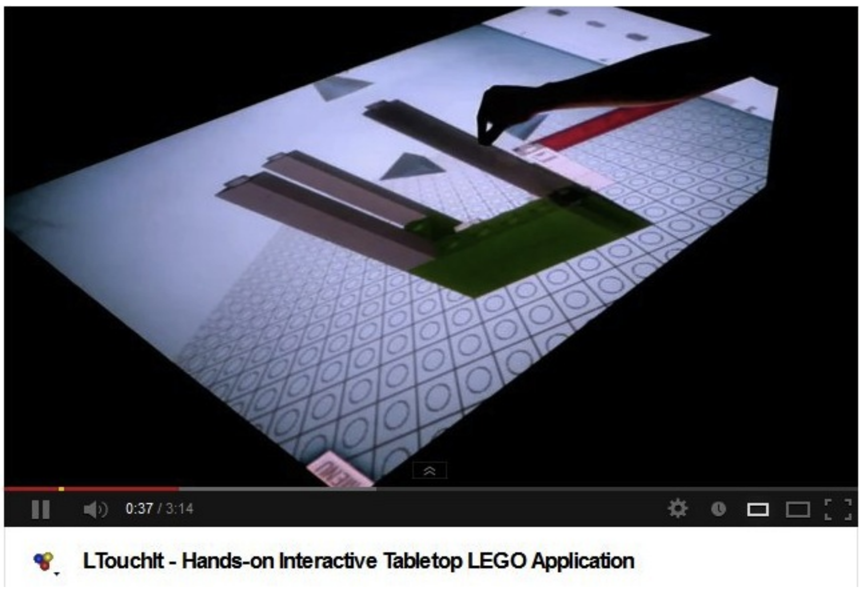


Figure 2.6: LTouchIt, a snapshot from a YouTube

2.2.2.6 Summary

In this second part of review, we could see that the same concept of using ”mode” applies to most of the interaction techniques. However, the interfaces discussed in this section were not limited to using readily available cheap devices only; there is a lot of work on natural interface developed through the use of tangible objects. Whilst the use of a tangible object is not much different from using the Wiimote - both devices need to be held during manipulation - the tangible object might require additional setup in order to make it trackable.

2.2.3 Other Work on 3D Natural Interface

In the rest of this survey is included work with a similar objective to ours, but with a significantly different approach. This includes work studying gesture recognition and work requiring special and possibly expensive hardware. Although it is not directly comparable with our work it is included because the ideas behind these contributions are important to help us understand what makes good natural interaction.

2.2.3.1 Wiimote and Gestures

The Wiimote has been used a lot in gesture-based applications. S. Sreedharan et al. [100] explored the possibility of using the controller to provide 3D gesture-based input to the 3D virtual world, Second Life. From its usability testing, the interactions were appealing and natural for gestures like waving but hard to map to facial expressions. Another gesture-based application that uses the Wiimote was done by E. L. Wong et al. [111] for an interactive music performance system on a PC platform. The controller was creatively used to perform conducting, percussion and sound manipulation.

T. Schlomer et al. [92] developed a system that allows arbitrary Wiimote gestures to be trained by using one hand. This gives the flexibility to the user to personalize their own preferred gesture for interaction rather than using a predefined set of gestures. To use the Wiimote as a pointer requires the user to face towards the sensor bar and this limitation doesn't make the controller suitable to be used in a "surround environment". T. Sko and H. Gardner [96] improved this limited interaction by introducing a Multiple Sensor Bar system which allows the Wiimote to be used in a two-walled, VR theatre.

2.2.3.2 Novel Interfaces

Some researchers prefer to create a novel TUI as an input. Y. Huang et al. [51] and Y. Huang and M. Eisenberg [50] invented Easigami, a reconfigurable system of thin flat polygon pieces that can be folded. The physical arrangement of the pieces, connected by electronically instrumented hinges, is read into a computer and displayed interactively in real time. X. Cao and R. Balakrishnan [20] demonstrated a variety of interaction techniques through the use of a passive wand tracked in 3D for large displays. Since the TUI, known as the VisionWand, is lacking buttons, the authors developed a set of postures and gestures to track its state and enable command input.

T. Doering et al. [27] created a working prototype TUI by physical composition where its physical interface elements were separated from its functional part. The concept was based on an inner Core where it provides the basic technical and software infrastructure of a product while an outer Shell allows for a flexible interface to be designed such as adding a button or a slider to the interface. S. Hunter et al. [52] developed educational applications software for children, Make a Riddle and TeleStory, on the Siftables platform [2, 78], which is a hybrid tangible-graphical user interface equipped with motion and neighbor sensing, graphical display and wireless communication. The platform provides children with responsive feedback to encourage manipulation and to increase engagement while playing with it.

G. W. Fitzmaurice et al. [29] developed a graspable user interface through the use of physical handles called "bricks" that can control the electronic or virtual objects directly. The bricks can be tightly coupled or fixed to virtual objects for manipulation or for expressing action. H. Ishii and B. Ullmer [54] proposed their vision, the Tangible Bits, that can integrate the augmented world with the physical environment. They implemented three prototype systems to explore the use of physical objects as a way of manipulation in the center of users' attention, as well as using ambient media as a way to communicate information at the periphery of human perception.

By using an eyedropper metaphor, J. Zigelbaum et al. [121] developed Slurp for location-based media that can extract digital media from physical objects with haptic and visual feedback. Slurp works where the media can be slurped up when being touched by the device, and then the content can be squirted out by touching it to a screen or pointing it at a remote display. J. Lee et al. [68] presented ZeroN, a tangible representation of a 3D coordinate of the virtual world that can be levitated and moved freely by computer by using a magnetic control system. This interface enables mid-air interaction that can be used in a lot of simulation such as architectural, physics and entertainment.

M. Fjeld et al. [30] presented an instantiation of NUI called BUILD-IT which allows multiple users to interact simultaneously in one common space to perform a design task for assembly lines and plants. The system uses a multi-brick interaction tool to render virtual objects where the bricks can be positioned and rotated according to their correct location.

P. Reuter et al. [90] developed ArcheoTUI that uses tangible props to manipulate virtual fragments of fractured archeological objects. These props are electromagnetically tracked, which can be mapped directly to the corresponding virtual fragments on the display. K. Y. Cheng et al. [22] described an approach where users can utilize an everyday object to be an instant tabletop controller. They developed iCon, a protoype platform to analyze the possibility of using an object from the current user's work environment as a TUI.

2.2.3.3 TUI and Direct Touch Tabletop

There is also work done to adopt the use of TUI with the tabletops. P. Dalsgaard and K. Halskov [25] proposed an interface that combines tangible interaction, tabletop interface and 3D projection. This tangible 3D tabletop enables a precise projection of content such as a blueprint of a building or information about cultural institutions or 3D building facades onto tangible objects placed on the table.

Collaborative virtual environment (CVE) supports collaboration in a virtual, 3D world where users would feel as if they were interacting in the real world. Following the CVE approach, A. Wu et al. [112] presented their work on navigating and manipulating objects by using tangible objects and a tabletop interface. The tangible controls are integrated with a top-down perspective which the authors claimed can provide the users with a better understanding of the virtual world and makes the interaction more natural.

M. Hancock et al. [41] explored the use of tangible and direct touch interfaces to accomplish a 2D information visualization exploration and 3D object manipulation tasks on a digital table. They found out that navigating the visualization was effectively done by using the tangible interaction whereas the touch interaction could move and rotate objects in 2D faster.

2.2.3.4 Hand Tracking Techniques

In another study, K. Hinckley et al. [46] proposed a work on a multiple DoF input with two hands to support 3D neurosurgical visualization as a post-WIMP interface. From the experimental evaluation, the result showed that the two-handed interaction can be twice as accurate as one-handed performance for the task of mimicking a posture without visual feedback.

Tracking hand gestures with a camera is also a popular choice among researchers. R. Held et al. [42] developed a system where puppeteers can use their own toys and props to directly perform 3D animations based on motion capture. These puppets are manipulated in front of Kinect camera where the system will use image-feature matching and 3D shape matching to recognize and track them.

B. Yoo et al. [116] proposed an interaction method using a combination of a hand and a line of sight (LoS) tracks by a far-distance depth camera. Their methods require the whole interaction screen to be divided into a set of interaction sub-regions. Based on these sub-regions, the LoS is used to select any sub-regions while the hand is used to point toward the region around the object of interest before manipulating the object. This interaction uses the concept where we naturally align our pointing hand with the LoS. S. Ghosh et al. [37] developed a realtime 3D markerless system that can detect and track multiple hands simultaneously. This system allows users to move freely in any direction and orientation and at the same time maintaining the information of each hand movement over a period of time.

R. Wang et al. [106] proposed another markerless bimanual hand tracking system for assembling a 3D object in 6DoF. The hands are tracked using two webcams. This system allows users to use the mouse and the keyboard if they need to, and automatically stop the hand tracking while the conventional interface is in use.

Y. K. Jin et al. [55] created GIA, a gesture-based interaction photo album for storing and managing digital images in a photo album. Users can choose an album in a pick-and-drop manner, manipulate the size and the orientation of the images with their finger or turn a page to view the photos. N. Henze et al. [44] developed a consistent set of free-hand gestures to control music playback after refining the gestures based on a constant user feedback. The gestures were used for common manipulation when playing music such as play, stop, pause, changing the volume and move to the next or previous track.

2.2.3.5 Mobile Devices

Besides the comodity hardware, mobile devices are also widely used in this area, either as a camera to track gestures or as a tangible input itself. H. N. Liang et al. [70] investigated the possibility to combine surface and motion gestures on mobile devices to perform 3D manipulation on large surfaces. Based on experiment, the authors implemented a potential interface, the SquareGrids, a single-sided multi-touch tablet that can be used to manipulate distant objects, with the aid of its accelerometer and gyroscope.

J. Zizka et al. [122] used motion-based interaction adopting a laser speckle sensing technique for their applications. They developed several prototypes. TouchController is a combination of remote translation and multitouch where it can act as a regular mouse, as well as being tracked in midair, with a multitouch surface. MobileViewport combined their 3D tracking technique with a mobile phone. They also developed a motion input for public displays where they integrated an optical mouse sensor and laser behind a glass window.

2.2.3.6 Virtual and Augmented Realities Environment

S. Neale et al. [79] used the concept of snapping pieces into place when they were correctly positioned in a 3D AR Puzzle application. The prototype allows users to physically orientate the virtual object to the right position and the snap interaction acts similar to the way the objects in the real world click when they fit together or the way two magnetic objects attract each other. In the Arch-Explore application developed by G. Bruder et al. [17], users can naturally explore 3D architectural models at different scales by walking into a VR environment using either a head-mounted display or CAVE setups. Since the interface registers physical with virtual walls, users are able to feel the passive haptic feedback when they touch some walls in the virtual world.

N. Petersen and D. Stricker [85] used a different way to interact with the virtual world. They enabled the switch of domains between the contents of the interface of a virtual instance and a physical instance in the real world. Based on a 3D hand gesture interface, a multi-touch interface and a paper-based object, the system works by first tracking the user's hand in 3D. The user can point to select parts of the displayed object, and then "grab" (pulling and move away) the information. The information can be visualized as a virtual sheet of paper held between the fingers.

2.2.3.7 Multitouch Displays

For applications on multi-touch displays/tabletops, S. Strothoff et al. [101] proposed a 3D interaction technique, Triangle Cursor, that produces 4DoF of object manipulation above the tabletop. The interaction uses an indirect approach where the users can select an object, position it, adjust its height and perform yaw rotation. This method can be further extended into 6DoF using a trackball metaphor to control the pitch and roll of the object.

Y. Yin and R. Davis [115] developed a recognition system that can continuously track 3D hand postures in real time with a tabletop display. The gesture recognition was tested with the Google Earth 3D map on a web page where the background image of the map on a display changed according to the gesture. A. Martinet et al. [73] explored a way to map finger gestures to 3D movements to position 3D objects on multi touch displays. They designed a Z-technique which allows 3D positioning to be done in a single view of the scene.

J. L. Reisman et al. [88] extended the principles used in 2D Rotate-Scale-Translate into 3D. They used one and two-finger interactions to manipulate the object in 4DoF as in 2D, and used three-finger interaction to create a complete 6DoF manipulation. These interactions are actually two-handed interaction. A. Sharma et al. [93] developed MozArt, a multimodal interface that combines touch and speech input to perform 3D modelling. The manipulation of objects is done on a tiltable multi-touch table whilst speech commands are used to perform actions that were previously done through the use of menus and icons.

F. Daiber et al. [24] evaluated the effect of position and motion parallex of indirect multi-touch 3D selection techniques under stereoscopic display. When head-coupled perspective and touch-based interaction are combined, this produces unintended object movement, thus a one-to-one correspondence can't be achieved. Y. Sugano et al. [102] attempted to solve this problem by creating a consistent perspective for direct interaction under motion parallex on multi-touch surfaces. This would increase the natural experience when viewing and controlling the 3D environment because the position of the projected image plane will be changed according to the head movement.

M. Hachet et al. [39] combined a multi-touch workspace with 3D stereoscopic objects and 2D monoscopic content where users can benefit from direct and indirect interaction. In this system, users can see the floating 3D object in the same space as the data displayed on the touchscreen.

2.2.3.8 Hybrid Approach

Other researchers who went for this hybrid approach were S. Baumgartner et al. [9]. To avoid wearable devices, the authors used a special 3D display device to project the stereoscopic display, and a Tablet PC acts as a 2D interface to perform tasks that don't need to be done in the virtual environment such as reading 2D text and browsing for information.

K. Hinckley et al. [48] also chose to combine two inputs to create a new tool. Based on pen and touch core tasks, users can perform different sets of gestures for each interface. For example, the pen is used for writing/drawing strokes, whereas the touch is used to manipulate interaction such as zooming, flipping pages, moving and selecting objects, and creating new objects. The authors's emphasis was not on evaluation; however they did test the different sets of gestures with 11 users, who found their approach to be appealing.

2.2.3.9 Pen / Stylus Input and Haptic Feedback

E. Ahn et al. [4] proposed a way to do 3D tracking for an auto stereoscopic display on a mobile phone by modifying a standard stylus into an articulated device with joint sensors. Feedback such as sound, visual and tactility are used to assist in selecting the object in the small phone space. A. Withana et al. [109] also used a stylus which incorporates a specialized haptic force feedback to enable direct touch and manipulation on a surface environment. ImpAct, as they called it, is a pen-shaped device that can change its length when it is pushed against a display surface, and appears inside the display device as a virtual stylus as if the stylus has penetrated into it. This enables the interaction with the virtual objects.

Still using a pen as an input, A. Agarawala and R. Balakrishnan [3] developed BumpTop, a virtual desktop prototype where users can organize their desktop by dragging or tossing the objects under the influence of physics. This means the users would experience friction and mass when organizing the desktop, and the objects can collide and displace others as in the real world.

The use of haptic feedback can increase the naturalness in the interaction. By focusing on this concept, B. Baxter et al. [10] developed DAB, a system for artists to draw in a virtual environment by using a 3D virtual brush. A. Rowe and L. Birtles [91] also opted to use haptic feedback to allow users to naturally interact with a virtual environment. With Glowing Pathfinder Bugs as objects, the creatures are aware of, and can respond to the change of environment surrounding them.

2.2.3.10 Indirect Interaction

An indirect user interface is a conventional way to interact with the computer. For this type of interaction a mouse is commonly used. There was work done in 1986 where the goal was to position a 3D object interactively. E. A. Bier [11] introduced the concept of Skitter and Jacks where the jacks are the coordinate frames used for 3D object transformations, whereas the skitter is a 3D cursor used to position jacks in the scene.

J. Y. Oh and W. Stuerzlinger [81] proposed a new technique to move objects in CAD by using the mouse. The basic process is to find a movement surface, which is mapped to the mouse movement and let the selected object slide on the surface as the mouse cursor moves. A. Khan et al. [59] introduced ViewCube, a 3D orientation controller and indicator to assist in the manipulation of a 3D object for CAD users. It contains 26 possible views for a 3D object where each view is mapped directly to the object so whenever the view changes, the object will be affected too.

2.2.3.11 Direct vs Indirect

C. Forlines et al. [31] did a comparison between direct touch and mouse input for unimanual and bimanual tasks on tabletop displays. They conluded that it was really difficult to use two mice in parallel which also affected the performance. The bimanual touch input gave a better performance, but, it decreased in accuracy rapidly over distance.

C. Muller-Tomfelde and C. Schremmer [77] also compared direct and indirect methods. However, rather than comparing which input was better, they intended to observe how the difference between the touch and mouse inputs affected the patterns of collaboration in achieving a shared goal.

2.2.4 Current Interfaces for 3D Modeling

Computers have been used a lot to assist in the creation or modification of a design. Examples of the current popular tools used for this purpose are Blender and FreeCAD for free software packages, and Maya and 3ds Max for proprietary software.

The fundamental principles for the interaction used to create and manipulate a 3D model in these software are basically the same. The only major differences might be in terms of the layout and the commands used.

Let's take look at the Blender interface for an example [Figure 2.7]. When a user starts the Blender application, a 'grid' environment is displayed, showing the view of its three basic objects: a cube, a light and a camera. This view can be changed to "top", "front", "right" and "camera". The user can also split this view into 4-view window so he can see what the object looks like from different views at the same time.

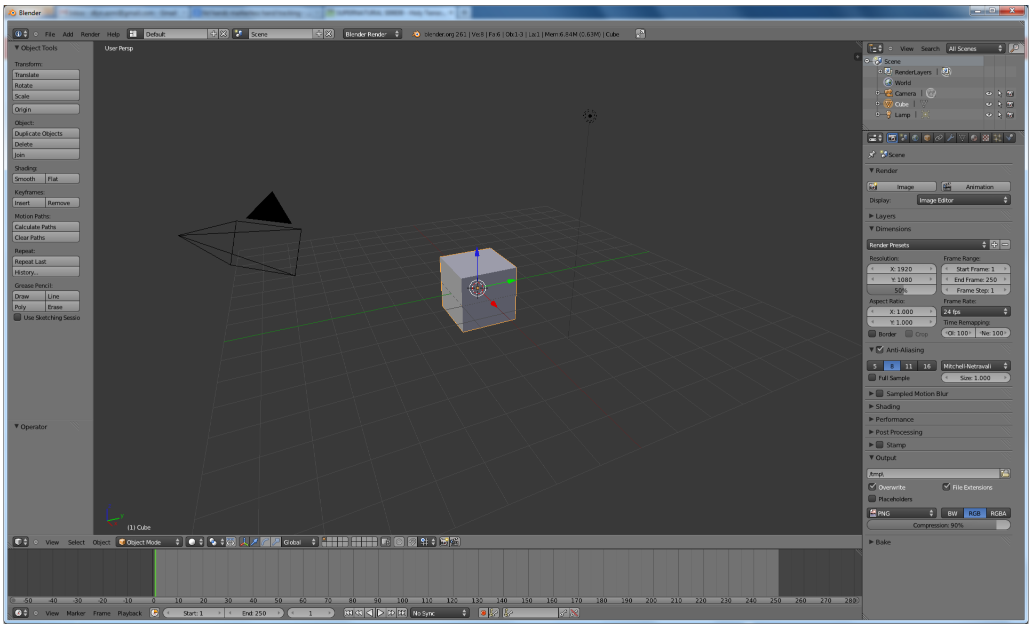


Figure 2.7: A Blender interface.

The interface uses several modes for a 3D manipulation. For example, to model or modify an object, the user has to be in an "edit" mode. To move around the object, the user has to switch to an "object" mode. To translate or rotate the object requires the user to use a different command, and the translation or rotation can only be performed one at a time.

This type of interface still uses a traditional mouse input to perform the task. Beginners will find it hard to master its interface without proper training. This interaction can be simplified if we allow the manipulation to be done in 6DoF. The user might still need to be in a different mode to differentiate between editing and moving. However the manipulation of the 3D object itself doesn't have to go through several steps just to displace the object in a slightly different direction.

2.2.5 User Interface Research at University of Otago

There has been work done on natural interaction at University of Otago itself. All of these applications were developed as part of Computer Graphics and Vision Lab's projects.

2.2.5.1 Watching Window

The Watching Window was the first natural interface project done here [75]. The idea was originated from Professor Geoff Wyvill, where he claimed:

> "Computers have looked so much the same for nearly twenty years. We are so used to the screen, keyboard and mouse that we forget that this appearance is merely a fashion and an accident of history. We should be able to communicate with a computer by speech and by gestures. Instead of using special tools like a keyboard, the computer can be programmed to determine our desires by watching and listening."

The basic concept of the Watching Window is to have a user being "watched" by two small cameras residing at each side of the screen [Figure 2.8]. From the head movement and the hand gestures, the computer must be able to deduce what the actions are. The most important concept for the Watching Window is for the computer to determine what the object will look like from the user's point of view.

The original Watching Window consists of a big screen and a box shape booth with white painted walls with two cameras. The current Watching Window uses a 42" LCD screen with the ability to show stereoscopic images.

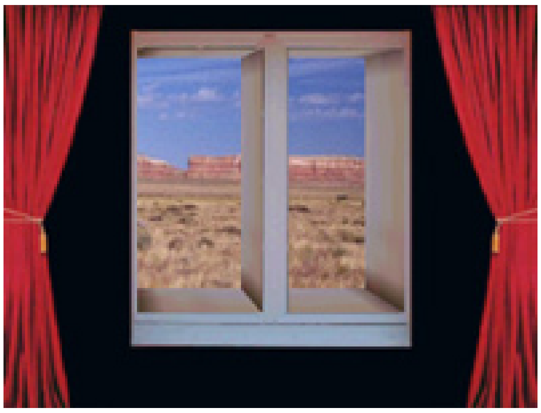


Figure 2.8: The Watching Window - the computer determines what the user will see.

2.2.5.2 Direct Manipulation Interface

In 2006, a student in our department, Natalie Zhao, wrote a Direct Manipulation Interface by using a mouse to manipulate a 3D object for her Master's thesis (N. Zhao [120]). A standard mouse is designed to work in a 2D environment. To allow the mouse to sense movement in all three axes is possible with some neat tricks.

If we look at the computer screen and at the same time we move the mouse straight to the right and then push the mouse straight ahead, we can see that the mouse cursor on the screen is moving to the right and then going up. The cursor will never go into the screen, doesn't matter how much we are pushing it forward.

By using the mouse behavior, she designed the desired directional movement. It is pretty obvious that if we are pushing it forward or backward in a straight direction, the cursor will only go up or down. This is where N. Zhao's idea came in. The user's intention can be predicted and we can use the knowledge to our advantage.

Based on the prediction of the user's intention, to move the object up will require them to push the mouse forward in straight direction. Therefore she concluded that if the movement forward or backward from one mouse point to the next is within a predetermined range, the object will be moved up or down. N. Zhao used 4° as the vertical range [Figure 2.9]. I couldn't justify the reason for using 4° because it wasn't mentioned in her thesis but I assumed the value is small enough to be considered as a straight movement.

The movement in the x-z plane (the horizontal plane) is considered to be the normal state motion. When the mouse is moved outside the range, it can be safely assumed that the user's intention is either to go horizontally or in z direction. This free motion to anywhere in space can be achieved by combining the vertical and horizontal movement of the mouse.

Our initial implementation of this method will be discussed more in Chapter 4.

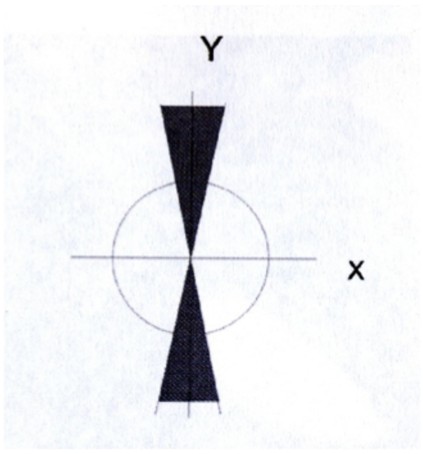


Figure 2.9: The X-Y plane where the shaded area represents the 'vertical' movement of the mouse.

2.2.5.3 The Octagon

The Octagon is an application that allows eight "artists" to draw a 3D sculpture in a shared virtual room [113]. For this purpose, eight computers are connected through a network. At the center of the virtual room, there is a platform for the users to build their sculpture [Figure 2.10]. Each user can add to the sculpture by selecting any available primitive shape in a very intuitive way.

A mouse is used as an input. The beauty of this application is, even though the mouse is a 2D tool, whatever shape being drawn on the platform will appear in 3D. Therefore, the users need not worry about the technical details. They also don't need any instruction on how to use the application. The concept is very direct and they will get the feel of the naturalness as soon as they grab the mouse and draw the first shape on the platform.

But the direct and easy nature of the application is an illusion. It is achieved by restricting the elements that can be drawn. What appears to be a 3D curve is actually restricted to a vertical plane. This does not give the full 3D control needed to make arbitrary shapes.

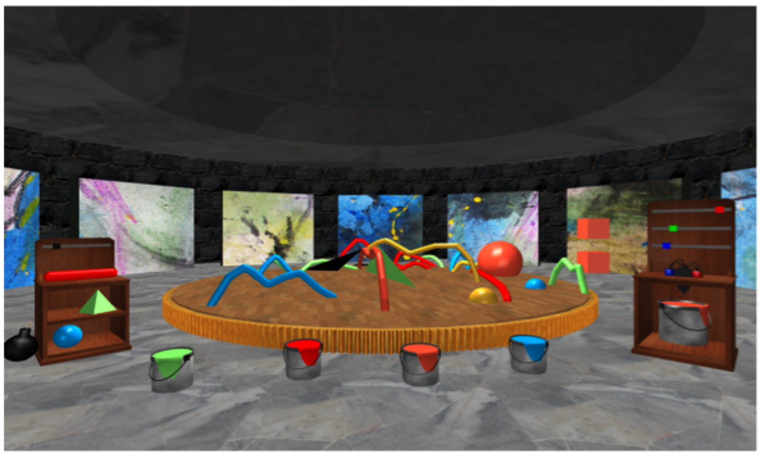


Figure 2.10: The Octagon's platform where the user can build their sculpture.

2.3 Summary

In this chapter, we reviewed several studies of 3D natural interfaces. We found many different styles of interaction, including TUI, hand tracking, multitouch tables, stylus and even mouse-based interaction. We compared the work with the characteristics of 3D natural interfaces as mentioned in Section 1.3: cheap hardware, 6DoF, 3D manipulation, testing by experiment and comparisons with traditional mouse-based interfaces. Interestingly, no single study has yet incorporated all of these characteristics. In our study, therefore, our goal was to incorporate all of them.

Our literature review also suggested some more specific directions for our natural interface research. Firstly, we decided not to use a mouse. The mouse is one of the most stable and popular interfaces created. As mentioned by L. Zhang et al. [119] the use of camera to manipulate a 3D object is unstable and less accurate than the mouse. The same can be said for other interfaces. But, using the mouse to perform a 3D interaction won't be straightforward. For instance, the direct 3D manipulation using a standard mouse proposed by N. Zhao [120]required several steps and a prediction to move a selected object to a specified destination. The system needs to be able to predict from the user's hand movement if the user is intended to move in the y- or z- direction. If the intended direction is along the y-axis, then the hand movement must always be within $4^o$of the vertical range, otherwise the movement is considered as moving along the z-axis.

Secondly, we decided not to use a camera. A camera is a popular choice for a natural interaction especially when we can get a cheap device such as the Kinect or a mobile phone easily([67], [64], [43]). Researchers can also opt to use cheap cameras to track hand gestures for manipulating an object ([119], [60]). The drawback of using the camera to track the movement is that it requires a set of gestures to be predefined. An unknown gesture might not be intepreted correctly and might lead to an error in the interaction. Another way is to use the camera to track a tangible object as demonstrated by D. Bradley and G. Roth [15] where they implemented a TUI system that can track a 6DoF pose of a sphere in a real-time video stream and then apply the pose to a virtual object. The restriction with this system is that the tangible object has to be within the camera's view and we don't know for sure how the latency would affect the interaction. O. Hilliges et al. [45] used a hand grasping method to interact with virtual objects that worked well. However, as we mentioned earlier, we want to avoid the complication of hand grasping, hence, for this purpose camera-related devices are not an option.

Thirdly, we decided not to use a multitouch table. A multitouch table allows a direct manipulation of the object, but the touch surface is 2D. This means a certain set of gestures have to be used to tell the computer to perform a 3D action. Furthermore, the multitouch device is not that cheap.

Fourthly, we decided to take physics seriously in our 3D interface. Not many papers described their 3D manipulation techniques using correct physics. Many of them just focused on how to perform the manipulation correctly and they also include scaling as part of the manipulation. For example, P. Song et al. [99] allowed scaling in their 3D manipulation task, and so did S. Han et al. [40]. Scaling of objects is useful for design tools but not a function that exists in the real world.

We do not want scaling in the manipulation because it is impossible to do in the real world. We are looking for an interface that can naturally manipulate a 3D object in 6DoF under the influence of physics as in the real world. Apart from correct physics, incorporating a visual cue such as shadows to give information about the depth of an object into the application is also a good approach. Some researchers implemented visual annotations to assist in the manipulation task. M. Chen et al. [21] implemented an assembly task that uses artificial cues such as a red overlay to denote if an object is graspable and a green overlay for the correctly aligned objects. M. Raj et al. [87] also used a cue in the form of a message prompt to tell the users if they had successfully matched the orientation of a test object with a given object. R. Wang et al. [106] did not perform their 3D assembly task any differently either. They also used visual annotations to differentiate each assembly action as shown in Figure 2.4 above. We believe with a correct combination of an input device, an interaction technique and a calibration method, this lack of naturalness can be addressed.

Finally, the application has to be tested by the intended users and compared with standard 3D modeling software. There is published work comparing the performance of a proposed interface with a standard mouse interface. In this case, they wanted to find out if their interaction was more natural than the mouse interface by identifying or observing users' preferences, as described in R. S. Yang et al. [114], L. Zhang et al. [119], A. N. Antle et al. [6], D. Mendes et al. [76] and C. Forlines et al. [31]. C. Muller-Tomfelde and C. Schremmer [77] compared the performance of a touch input and a mouse input not to find which one is better but to observe how the difference between these two could affect the pattern of collaboration in achieving a shared goal. L. Chittaro and R. Sioni [23] wanted to investigate the physiological effects of an LPS interaction technique vs a mouse and keyboard setup.

There are also some important concepts that we can use from the papers reviewed. Our focus is on a direct approach in manipulating the 3D object without any constrained movements. What we learned from M. Resl et al. [89] is that in order to create a natural and easy to use interface, the interaction needs to be direct and simple. That is exactly what we aimed to do.

We agree with Raj et al. [87]'s conclusion that the individual differences in users' spatial abilities and experience should be taken into account in designing the user interface. There is no point in creating an interface that can't be used by a majority of users.

In the next chapter, we present a conceptual model of our interface that uses a 3D input device, a one-to-one correspondence technique and a simple calibration method to create an acceptable 3D natural interface.

Chapter 3

Conceptual Design

In this chapter, we motivate the core design of our new 3D interface in Section 3.1, and describe the design itself in Section 3.2. In Section 3.3 we introduce a novel task to test the interface, and in Section 3.4 we describe how we propose to evaluate the interface in relation to other 3D interfaces.

3.1 Understanding The Problem

Our objective was to create an application where users are able to manipulate a 3D object as if they were playing with it in the real world. We chose an assembly task because so many jobs in the real world require assembling of objects.

We know that most current 3D modelling software used for assembling 3D objects is not natural. Although the environment is 3D, the interaction itself is 2D. As described in Subsection 2.2.4, each translation and orientation can only happen in one plane at a time. This type of interface is very hard for most beginners. That is why we need to develop a natural interface to allow users to perform the task more easily and faster.

However, by just simply creating something that has not already been done by other people alone is not enough. We choose to do something that we believe is better or addresses a new problem that we see as important for other reasons. Although the use of hand grasping can be really natural, we decided not to use this technique because it is so complicated to model. We opt to go for a simpler solution. Having an object that mimics the motion of a tangible interface object, on the other hand, is not as natural as having an avatar that represents the object in our hand.

How would our 3D manipulation be? We do not feel that using artificial visual cues is natural, and we plan to omit the scaling of objects as part of our manipulation since scaling is not something that we normally do in the real world. We need to figure out a way to allow the users to perform the 3D manipulation task without using artificial visual cues for them to know that they have completed the job.

3.2 A Proposed Solution: An Avatar which Acts like a Magnet

The natural way of performing this task is to use a hand grasping technique. However, as we mentioned above, simulating this type of gesture is too complicated. We need to design another method that can move and place a virtual object accurately and at the same time can avoid the complication of hand grasping. This interaction, represented by a virtual controller known as an avatar, is controlled by the user's hand gestures, and is tracked with 6DoF.

The term avatar here might be a little misleading since the most general and accepted meaning of an avatar is always associated with 'playable character' in the form of a human body [63]. However, according to Kadri et al. [56]:

> "Today, the avatars or 3D cursors used to display the user during manipulation task in VE can look very different. Indeed, the virtual shape of the user's avatar can be either a tool (e.g., screwdriver, hammer), the whole or subpart of the body (eye, hand, finger) or even any other object with or without a semantic content (arrow, star, sphere)."

Lok et al. [72] extended the defition of an avatar to include *"a virtual representation of any real object, including the participant. These real-object avatars are registered with, and ideally have the same shape, appearance and motion, as the real object."*

In our context, we will use the definition of an avatar as defined by Lok et al. This is also to make the meaning of our avatar consistent with the use of the term "avatar" in P. Song et al. [99], M. Raj et al. [87] and R. Francese et al. [32].

In Section 2.2.1 - 2.2.3, we described various ways of performing a 3D manipulation on natural interfaces. For natural interfaces, the method varies based on the interpretation of each researcher. It could be a hand grasping technique or tracking a tangible input or using a touch interface. It might not be easy to pick one typical interaction for natural interfaces. However, we could say that there is one common method used in performing a 3D manipulation task for the current standard modeling software, which is using a mouse in a constrained movement. How can we design a better interaction without restricting the movement of the avatar in space?

One technique that can fulfill this requirement is to use magnetic force. Magnetic force attracts iron and steel objects easily. If the avatar is modelled as being magnetic, and the objects are treated as if they were made of iron or steel, users can select which object to grab by moving the avatar closer to it. If any objects are within the range of the magnetic force, they will be attracted to the avatar [Figure 3.1].

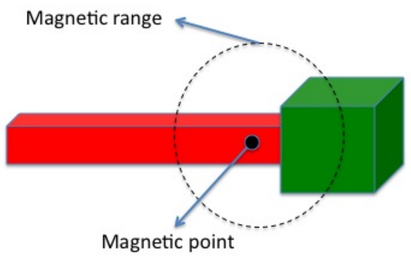


Figure 3.1: A cube within range will be attracted to a magnet.

The magnet in this context is similar in concept to drag in 2D. From the papers reviewed in Section 2.2.1 - 2.2.3,, we did not find any method similar to ours. The only magnetic metaphor found was used as a snap technique to fit two pieces of puzzle that were correctly positioned, as demonstrated by S. Neale et al. [79].

We understand completely that this technique could never replace the naturalness of hand grasping and will not be a choice to perform the assembly of objects in the real world. But for the purpose of this project, manipulating 3D object by using magnetic force can be used as a compromise. Apart from that, our technique can also simulate the softness of picking and manipulating the object as one would experience with hand grasping. This magnetic metaphor is described in more detail in Section 4.2.3.1.

3.3 A Task to Test the Proposed Interface

Having proposed a type of interface for manipulating 3D objects, we designed a task to make use of this interface and test it. To demonstrate the idea, we chose to simulate the task of putting cubes into tubes. This interaction requires users to grab a cube and place it in the tube by using 6DOF without any constraint.

If we refer back to Section 2.2.1, we can see how the other researchers evaluated the completion of their 3D manipulation tasks. Most of them use visual cues to denote that the 3D manipulation task had been successfully performed. For example, M. Chen et al. [21] used a green overlay for objects that were correctly aligned. R. Wang et al. [106] also gave every assembly action a distinct visual annotations such as a golden halo for an object selected for manipulation and a bounding box around the scene and a 3D manipulator for camera rotation and translation. M. Raj et al. [87] used a threshold value where the orientation of a manipulated object was considered a match if it was within 15^o of the orientation of the target object. A "match detected" message would be prompted to mark the completion. B. Froehlich et al. [33] used spines around the vertices of an object to indicate a docking tolerance. D. Mendes et al. [76] on the other hand, would just need to place a 3D object in a correct position and orientation, and the exact pose of the object would be "auto-corrected" by the system.

Our proposed interaction technique deliberately omitted explicit visual annotations to indicate success in the 3D manipulation task. Rather we were focusing on a more natural way, which uses physics to achieve the desired interaction. Users can employ their common-sense knowledge about the conditions under which an object is stably placed within a container to determine when the manipulation task has been successfully achieved.

We needed to answer some questions before we could create an application that can test our claim.

- How to represent the action in the virtual world?
- How big are the cubes compared with the tubes?
- Will the interaction be straight forward?
- How many times do the users have to repeat the task?
- What is the range of the interaction?

- What about the physics of the virtual world? How accurate should it be?
- Since users are working in a virtual 3D world, how can the users tell exactly where they are in 3D space?

3.3.1 The Action

In order to perform the task, users must be able to see their actions clearly. We decided to use a 3D graphical representation of the real world object as an avatar, to meet this requirement.

With the use of the avatar, users can move freely in the virtual world. They can see where they are and can determine what type of action to perform based on the position and orientation of the avatar.

3.3.2 The Design of The Scene

The idea is to build a room which contains several cubes and tubes. The size of the hole for each tube should not be too big, just enough for the cube to be placed in [Figure 3.2]. The shape of the hole is a square with rounded corners. This was done so that the users would be forced to use a 6DoF interaction, which means the users would have to carefully orientate the cube until it fits the hole nicely before it can be pushed into the tube.

The shape and orientation of the tubes demand a 6DoF manipulation and they are not aligned with any axis. All the tubes are oriented roughly about 45° facing the front view so that users can see the hole. To make interaction more interesting, we assign the same color for each cube-tube pair. To complete the task, users must put each cube into its matching tube. The size of the scene will be dependent on the size of the window created to display the virtual world.

3.3.3 The Interaction

The aim is to maintain the element of naturalness in the interface. Therefore, we apply the use of magnet as it is without any modification. Once the cube has been picked up, users can move the cube to the correct tube and place it accurately into the hole. The cube can be disengaged from the magnetic grip by sliding the avatar on any static (immovable) object in the scene or by shaking the avatar with force greater than the magnet. We make sure that the task is not that simple to perform, and this can be

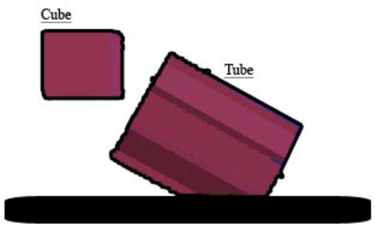


Figure 3.2: Placing a cube into a tube.

referred to the training video provided in Appendix B: Videos 3 - Instruction for the Experiment.

3.3.4 The Completed Task

The task is considered complete if all of the cubes have been placed accurately in the right tubes. To place only one or two cubes will be too simple so we decided to have four cubes that can be put into four tubes.

3.3.5 The Working Volume

One important aspect in this interaction is the range the users can work with. Since the task is about manipulating 3D objects in a virtual world, there must be a predefined range. It will not be natural if we have the virtual world displayed on a screen but the interaction can happen anywhere.

We designed a 3D working volume that is right in front of the virtual world [Figure 3.3]. All interactions are done within this working volume. This is to match our definition of naturalness where we want to achieve a one-to-one correspondence. The users must be able to see their simulated action exactly as if it happened right in front of their eyes.

3.3.6 The Physics

There won't be any naturalness if the physics is not correct. When the cube falls, it must have the right speed as in the real world. When two objects collide, the collision response should calculate the right behavior. Since the cubes are rigid objects, the

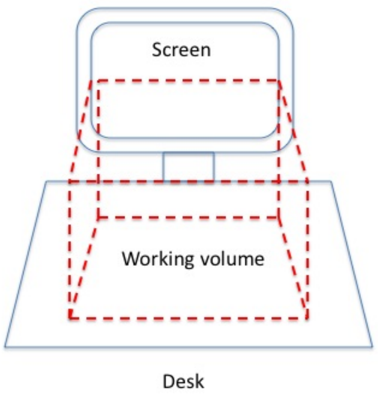


Figure 3.3: Working volume (in dashed-line) sits above the rectangle.

physics is more complicated and we have to consider all attributes associated with rigid bodies such as mass, friction, elasticity and damping.

3.3.7 Spatial Information

One of the problems faced by most people was to determine their location in the virtual 3D space. The issue occurs after they have selected the cube and try to place it on another object. They always find the selected cube to be in a different position as they get closer to the other object.

This is because it is very hard to tell the depth of each object in the 3D space. We can avoid this problem by applying shadow to every object in the scene. Shadows will give enough information about the position of the object in the 3D space as well as the distance from one object to another.

3.4 Evaluating a Natural Interface

The evaluation of our interface will have two central components. Firstly, we need to know how easy it is to use our interface with very few instructions. If it is 'natural', users should already have the skills needed to use it. Consequently, in our evaluation, we will be interested in users' very first experiences with the interface.

Secondly, we want to make an objective comparison between our natural interface and a standard 3D modelling interface. We want to demonstrate that our system improves over existing 3D interfaces, so this comparison is important.

3.5 Summary

We started by identifying the core characteristics in designing a natural interface and proposing a possible solution to perform a 3D assembly task in a natural way as described in Section 3.2. We explained why simulating a hand grasping is complicated. We decided to use an avatar which acts like a magnet to perform the assembly of objects. Although this technique might not be the choice in the real world, for this purpose it seems to be a good compromise. In Section 3.3, we defined a task based on a set of questions which covers the representation of the simulated world and its interaction and we decided to:

- apply correct physics as part of the natural interaction.
- use a 'kinematic' object as a means to control the virtual object. A kinematic object is not affected by the physics.
- calibrate our system particularly for the manipulation purpose.
- test our system without giving any instruction.
- compare our interface by using the same setup but different approach.

In the next chapter, we explain how this concept can be physically implemented.

Chapter 4

Physical Design

We developed an application based on ideas discussed in Chapter 3. We started by identifying the hardware and software needed to implement the interface.

4.1 The Requirements

We needed to select our hardware and specify the software. To build a house, we need to have the right tools such as the measuring tools, plumbing and saws. We need to decide what material to use depending on the type of the house - wooden or brick.

The same concept applies in order to develop our natural interface. Since we didn't intend to design our own hardware, we looked for something cheap, commercially available, genuine 6DoF, fast and wireless. We also sought software that complements our chosen hardware.

4.1.1 Hardware for Input

Finding the suitable hardware that can meet our objective is not an easy task. Some hardware might perform better in one application but fail in another. The only information we can get about the hardware is based on what is being advertised. Most companies claim their hardware to be of great accuracy. We can't, unfortunately, trust these claims and we had to make our own tests to discover what each device could do.

Reports and reviews written by other researchers and users were also helpful in understanding the hardware. Although they were using the hardware for a different purpose, the information provided was useful to perform the initial selection of suitable hardware.

4.1.1.1 The Mouse

The mouse is still a popular choice of input, from when it was first invented until today. Even with the invention of a new sophisticated input especially in the natural interface area, the mouse is still a choice of input for some people.

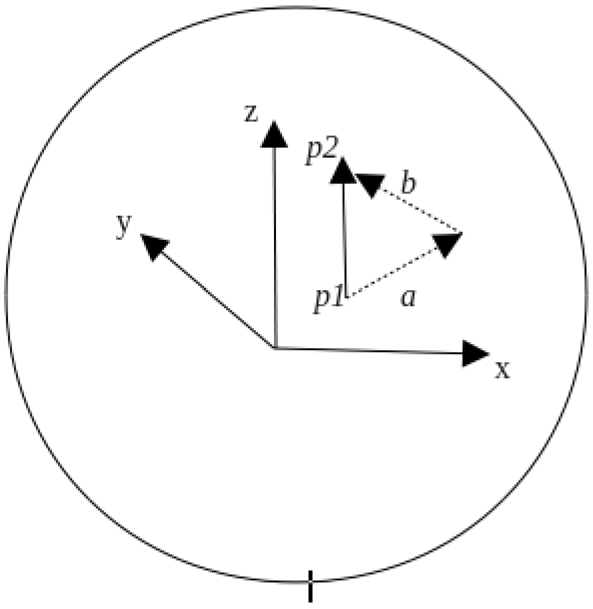


Figure 4.1: If the user wants to move from $p1$ to $p2$ which lie in the x-z plane, he has to take two steps: to move in the a direction, then in the b direction, to avoid accidentally moving the object in the y-direction.

We reviewed some of the techniques in the 80s and early 2000s that attempted to use the mouse to perform a 3D task in Section 2.2.2.4 and Section 2.2.3.10. To summarize that review: D. Venolia [105] created a 3D cursor that used a 'tail-dragging' metaphor to drag and rotate an object. E. A. Bier [11] also developed a 3D cursor to position a 3D object in scene, within a specified coordinate frame. G. Smith et al. [97] defined three different kinds of contraints to restrict the motion of objects in the virtual scene: unconstrained, partially constrained and fully constrained. They were compared with each other, and unconstrained seemed to be the slowest among the three types. J. Y. Oh and W. Stuerzlinger [81] implemented a different approach to orthogonal constraint to move an object with the mouse. The authors defined a movement surface that can be mapped to the mouse movement and let the selected object slide on the surface as the mouse cursor moves.

The use of the mouse to manipulate a 3D object also captured our interest. Before focusing on the 3D input, we attempted to implement the 3D interaction by using the standard mouse based on N. Zhao [120] directional translation method. As explained in Subsection 2.2.5.2 in Chapter 2, the movement of the mouse can be extended into the third dimension by predicting the user's intention.

Figure 4.1 describes our implementation according to this method. The circle represents the mouse movement area. The x-z plane lies on the circle area while the y-axis points up.

If the user's intention is to move from $p1$ to $p2$ in the z-direction, he will have to move the mouse in two steps as shown by a and b. In directional translation method, a straight vertical movement means a translation in y-direction which is pointing up. Therefore this additional step is necessary because the user needs to explicitly tell the system that the intention is to move on the same plane.

To move in a circle will require more additional movement. Since the user needs to avoid the movement being mistakenly interpreted as moving up, he has to move the mouse in a careful zigzag manner just to achieve the desired result. Figure 4.2 shows an example of a circle movement in the x-y plane.

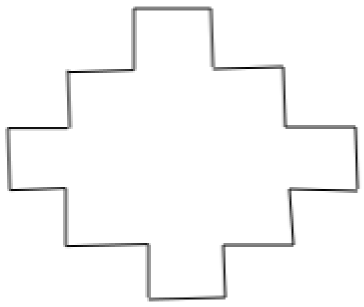


Figure 4.2: An example of a moving pattern for a circle in the x-y plane.

From this initial implementation, we can conclude that a standard mouse is not, and won't be suitable for handling a 3D object manipulation. What we just discussed is focusing purely on translation. If we were to incorporate rotational movement in this method, the interaction will get more complicated, and hence, defeats the purpose of the natural interface.

4.1.1.2 The Wiimote

The Wiimote was actually our first choice of a 3D input device. It has been successfully reverse engineered and information about this controller can be easily obtained from the net. There are also a lot of forums available for people to get help on any Wiimote related issues.

C.A. Wingrave et al. [108] described the Wiimote in detail. The Wiimote provides three axes of acceleration data. The data is not specified in any particular frame of reference (FOR) which means the Wiimote's spatial data is not directly mapped to the real world position.

The first FOR refers to the Wiimote's own x, y and z axes along with its pitch, yaw and roll orientation. In this FOR, $+x$ is pointing to the left, $+z$ is in the up direction and the head of the Wiimote is looking down the -y axis [Figure 4.3]. The second FOR is relative to the Earth since the Wiimote's accelerometer detects the Earth's gravity. The third FOR describes the Wiimote relationship with the sensor bar.

However, the data returned by the accelerometers is very noisy. Calculating the position of the Wiimote by double integration of the acceleration will not produce an accurate result. Most people apply a Kalman Filter to smooth out the reading. And since the Wiimote z-axis is aligned with the Earth's gravity vector, determining yaw in the Earth's FOR is also not possible.

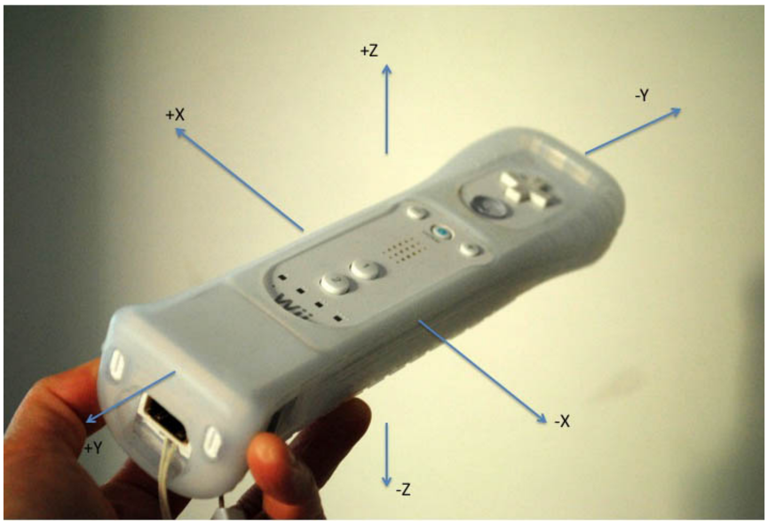


Figure 4.3: The Wiimote showing all three axes

Apart from the accelerometer reading, data can also be obtained from the optical sensor bar. The sensor bar emits infrared light that can be picked up by the Wiimote's camera. Based on the infrared reading, the x and y positions and the depth information can be determined. All of these data can be accessed through the use of a Wii library.

I installed the Wiiuse library written by Michael Laforest to link my application to the Wiimote. What I discovered from the initial implementation was that the controller is not a true 6DOF input.

As explained before, calculating yaw is not possible. In order to translate the avatar in the x-axis, we had to roll about the y-axis while moving our hand horizontally. This type of action works well with gesture-based applications such as tennis and baseball games, but not for natural mapping.

To get the absolute position, we can read the data from the sensor bar. However, this requires the Wiimote to be pointed at the screen at all times. Once the controller points away from the sensor bar, the connection will be lost and the data will be gone.

Although the Wiimote has been succesfully used in a lot of research study, this is the one area that it can't conquer.

4.1.1.3 The PS Move

The PS Move works differently from the Wiimote in the sense that it uses a camera, the PS Eye, to track the controller's motion. The controller, designed in the shape of a wand, has a sphere with RGB LED attached to its head [Figure 4.4]. The tracking of the wand works by finding the sphere in the camera's image. Once found, the size and location of the projected sphere can be used as a starting point and mapped to a 3D model.

The great advantage of the PS Move is that it can be manipulated in 6DOF. Besides, Sony itself has released Move.me software that can read and return the controller's data from a machine [74]. The software is designed for academic researchers as well as human-computer interface enthusiasts. Unlike the free Wii libraries, this software comes with company's support if users faced any problem while using the system.

However, this software also serves as the major weakness for most people. Move.me can only be purchased and downloaded by those who reside in the North America region. This is the reason why the hardware was abandoned in the first place. We needed to go through a lot of steps in order to get to the right person, and the whole process took months. By the time we finally managed to download the software, I already had my application working with the Hydra controller (see Subsection 4.1.1.6).

I tried the demo application that came with the software. Based on my observation,

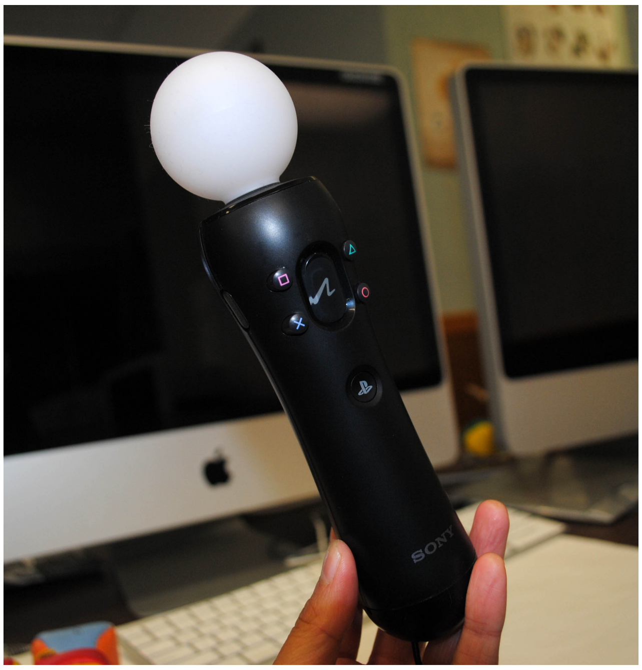


Figure 4.4: PS Move's motion controller with a sphere attached for tracking.

the orientation data read from the accelerometer and the gyroscope seemed to be very accurate because the virtual object imitated the orientation of the actual wand exactly as it is. There was a little latency when translating the wand. This is quite expected since the information was given at camera rate. We did not pursue the accuracy of the PS Move in detail since we had already chosen to focus on the Hydra controller. However, this is not enough to conclude that the PS Move is not suitable for this application. This controller will be a good test to be implemented into our application as future work. A detailed review of this controller can be found in [83, 104].

4.1.1.4 The Leap Motion

The Leap Motion controller is one of the latest game devices. It was first shipped in July 2013 for Mac and PC users. According to the manufacturer, this controller can track both hands and all fingers simultaneously with great precision, within one hundredth of a millimeter, and a latency lower than the refresh time of a typical screen [38].

The controller is a small device (79x30x11mm) with a glossy black panel on top, covering the infrared sensors [Figure 4.5]. The leap differs from the Kinect in the way it works using infrared optics and cameras. It tracks more accurately than the Kinect but over a much smaller range.

Figure 4.5: Leap Motion controller.

Since we had completed our study by the time this controller was released in the market, we did not have the opportunity to test the device. However, this latest innovation is definitely worth trying in the future.

4.1.1.5 Expensive 6DoF Input Devices

There are 6DOF input devices available on the market that have proven to be accurate such as the Polhemus and the Flock of Birds. These two are the examples of hardware that use trackers attached to a body to record the body movement. However these cost thousands of dollars. Even if we managed to get one, this project might not be able to be carried out by other researchers due to the high cost. Hence this will also defeat our purpose to ensure continuity in this project.

4.1.1.6 The Razer Hydra

The Razer Hydra controller may not be known by many people except for Windows gamers. The reason might be because it works only with the Windows operating system which makes it less visible to others.

The Hydra consists of motion sensing controllers, base station and cable to connect the hardware [Figure 4.6]. The base station uses a very low-power magnetic field to calculate the position and orientation of the controllers.

What is great about this controller is that it is a true 6DOF input. The manufacturer claims that tracking is accurate to 1mm and orientation to 1^o. Testing the

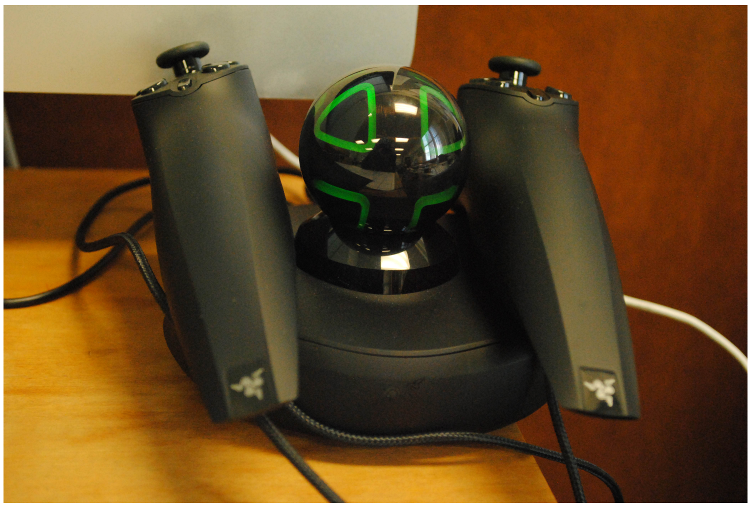


Figure 4.6: The Razer Hydra - consists of a base station and two motion sensing controllers.

controller is explained in Subsection 4.2.2 whilst calibrating the controller is described in detail in Chapter 5.

Unlike the other controllers that use accelerometers, gyros, LEDs and cameras to detect motion, the Hydra doesn't need to be calibrated frequently during use. It also doesn't require a line of sight to operate.

Another important feature of this controller is that it provides a low latency feedback. The response is really fast and it is almost impossible to detect any delay in its motion. This result is achieved through its magnetic motion tracking technology.

The cost for a complete Hydra set is about USD$140 which is not too expensive. As mentioned earlier, our ideal hardware should be cheap, off the shelf, genuine 6DoF, fast and wireless. The Hydra controller is all of these except wireless, which makes it the most suitable hardware for the project.

4.1.2 Software Support

To develop my application I had to choose an operating system, language and development systems, link to the drivers for the Razer Hydra and incorporate a "physics engine". Here I descibe the choices of these components.

4.1.2.1 The Operating System

To be able to work with the Hydra, Windows operating system is necessary. This is because the controller is specifically built only for PC users.

4.1.2.2 The Language

The choice of language has always been C++. This language gives users the flexibility to structure their own programming paradigm, from procedural to functional to object-oriented programming.

4.1.2.3 The Editor

Since I'm working with the Windows operating system, Microsoft Visual Studio seems to be the perfect choice. It is an integrated development environment (IDE) from Microsoft and can be used to create many types of application, console or graphical user interface. The version I'm using is Visual Studio 10 (via Visual C++).

4.1.2.4 The Physics Engine

As mentioned in Chapter 3, physics is an important element in developing a natural environment. There has been a lot of research published on how to simulate mechanics such as in D. Baraff [7], J. Gascon et al. [35], D. M. Kaufman et al. [57] and D. M. Kaufman et al. [58]. Physics engines that perform this function are also available.

The original idea was to develop my own physics engine. However, after a few experiments it became clear that this was too big a job and since the objective of this project is to create a natural user interface and not to develop a new physics engine, we decided to use an available engine.

I did some review on free physics engines and found out that Bullet engine is at the top of people's choice. Bullet is an open source physics engine used by a lot of professional developers and can be downloaded from [1]. Among popular projects created by using the Bullet engine are:

- Commercial Games
    - Toy Story 3: The Video Game
    - HotWheels: Battle Force 5
- Movies

  - 2012 by Sony Pictures Imageworks
  - Megamind and Shrek 4 by PDI/Dreamworks

- 3D Authoring Tools
  - Blender
  - LightWave 3D CORE

Apart from the above, the engine itself has been integrated into many open source game engines such as Panda3D, Ogre and Irrlicht Engine. These examples give an initial justification to Bullet's capability as a great physics engine [1].

Before integrating Bullet into Visual C++, I looked for the possibility of using the Blender Game Engine (BGE) because it uses Bullet physics together with its game engine. BGE uses the concept of "logic bricks" to control the movement of objects. "Logic bricks" works by combining three elements known as "sensors", "controllers" and "actuators".

To perform a more complicated action, Python script can be used together with the game engine. Once you grasp the concept of "logic bricks", creating a 3D game with BGE will be a simple task. It also provides support for joystick and a lot of gamers already integrated the Hydra controller with BGE as demonstrated in https://www.youtube.com/watch sI371le4. I chose to work with the C++ evironment because I needed close control of the simulated physics.

4.1.2.5 The Hardware SDK

Sixense has made it possible for developers and researchers to create games using all Sixense-powered controllers which include the Razer Hydra. The Sixense SDK provides libraries that allows the 6DOF motion tracking of position (X, Y, Z) and orientation (pitch, roll, yaw) to be read from the controller.

4.2 Implementing a Natural Interface with The Hydra

As explained in Subsection 4.1.1.6, the Hydra controller makes a perfect choice for our application since it provides true 6DOF input with a very fast response. It is also cheap enough that almost everybody can use it for their application.

However, there are also bad sides to the Hydra that needed to be taken care of before we can come out with a complete natural interaction. There were three crucial aspects to be corrected.

First, we already established an understanding that the Hydra uses magnetic fields to calculate the position and orientation of the controllers. This creates problems whenever there are iron and steel objects near the controller.

We did not know where the center of rotation of the controller is. This does affect the simulation of the avatar. Apart from that, the raw coordinates returned by the controller are non-linear. It is hard to tell where the points actually are in space. Another minor disadvantage of the Hydra is that it is wired which makes the movement a little bit encumbered.

4.2.1 Designing the Scene

The first thing we did was to design our 3D world. Without the simulated world, we could not create the interaction. Our world consists of a set of cubes and tubes which are lying around in a box.

To avoid inconsistencies, we used a meter unit for all the measurement. Since all of the objects should be visible within the display, their size should not be too big or too small. For this purpose, we defined a $0.04 \times 0.04 \times 0.04$ m as the size of the cube. With a set of four cubes and four tubes, this size occupies the screen evenly.

To represent the Hydra controller, the avatar was drawn as a compound shape made of a cube and a cone. To develop a natural feeling when manipulating the 3D objects, the avatar has to be the same size as the actual controller.

However, Bullet doesn't work well with very small or very large numbers. The range is between 0.2 and 5 unit. When I used a very small value such as 0.04 m as the parameter for the height to draw a cone, which is outside the range, an almost spherical polyhedron was drawn instead (Figure 4.7a). To fix this problem, we applied a scaling factor to every object so that the objects are big enough to be seen in the scene and the avatar looks identical to the actual controller.

So we are representing a 4cm cube by a 4m cube. But this affects the way it falls. The acceleration due to gravity is 9.81 ms^{-2} and in one second, an object falls about 5m from rest. So a 4cm cube falls 125 times its height while a 4m cube falls only 1.25 times its height. To make the larger cubes act like smaller ones we have to scale the gravity constant too. With $g = 980$, a 4m cube falls 125 times its height in one second so its motion looks like the motion of the 4cm cube. Figure 4.7b shows the complete scene.

Another factor that needed to be considered is the movement of the avatar. In the real world, we move the controller with our hand so we have complete control of the object. In the virtual world, the object can be static or dynamic. Bullet has its own definition of these kinds of object.

A static object is an object defined with zero mass and it doesn't move at all. A dynamic object is controlled by the simulated physics. If we were to build the avatar as a dynamic object, the movement of the avatar will not be that smooth because gravity would be pulling it down.

An object in Bullet can also be defined as kinematic. A kinematic object can be moved but it is not subject to the simulated physics. In Bullet, a kinematic object doesn't collide with any other static objects and can go through them, but it does collide with dynamic objects.

We want the avatar to move with the hand and interact with the dynamic objects, but without haptics we can't stop the movement of our hand whenever the avatar collides with a static object. Thus we will lose the one-to-one correspondence of the interaction. For this reason, we represented the the avatar with some transparency to avoid a non-realistic simulation if we were to allow the "solid" avatar to go through other static objects.

From Figure 4.7b we can see that the room has only three walls. The front part of the room is built without a wall which leaves a high possibility for the cubes to go out of the room and fall down due to gravity.

The first idea was to build an invisible plane to prevent the cubes from moving out of the scene. But that would make the simulation less natural whenever the user moves the controller out of the working area and the cubes just stop at the front of the room.

We chose to replace the cubes back into the scene whenever they went outside boundary. This approach also makes the simulation less natural. However, if the cube falls out of the scene while completing the task, it can be quite annoying. Therefore this step is necessary to compensate for the lack of haptic feedback in this application.

4.2.2 Initial Testing

With the completed scene, we proceeded with the integration of the controller to the application. We needed to check the operation of the Hydra controller. In particular:

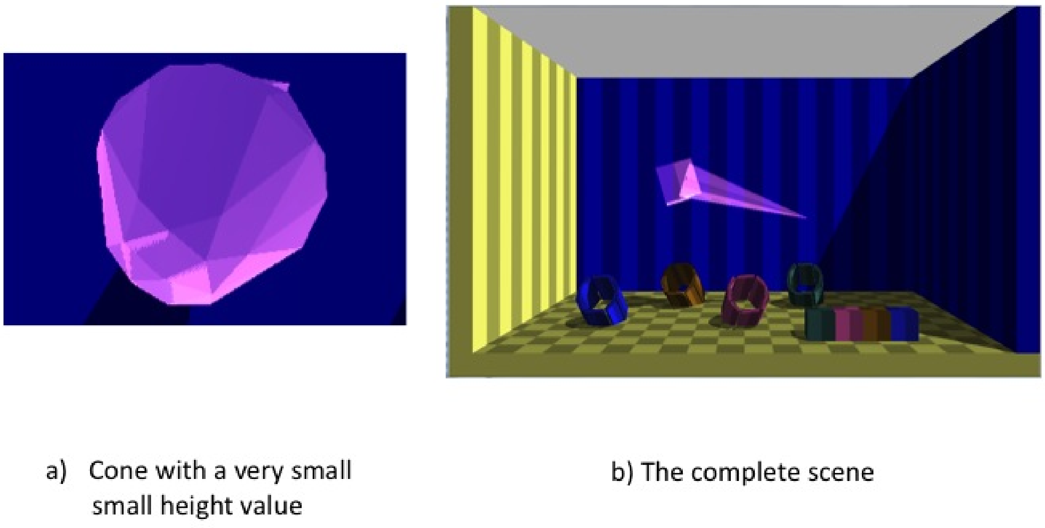


Figure 4.7: a)A cone with a very small height value. b) The complete scene.

- To see if its position reported depends on its orientation.
- To see if the orientation reported depends on its position.
- To measure and correct scaling and distortion of reported position. This calibration process is described in Chapter 5.

4.2.2.1 Getting Rid of Magnetic Disturbance

When we first implemented the Hydra into the application, we noticed something was not quite right with its movement. It did not behave according to the real world action. A straight horizontal movement was simulated as a diagonal arc. A movement in and out along the z-axis resulted in a curve motion.

Before putting the blame on the Hydra, we did some analysis to discover at which position the controller started to behave erratically. The cause of the problem was steel objects that disturbed the magnetic field.

The desk I was using had a steel frame. The chair I was sitting on also has some metal parts. When the controller was brought closer to these steel objects, the avatar started to behave differently. Even if the controller was put statically on the desk and a strong steel object was placed right near to it, we could see that the simulated avatar was moved due the magnetic attraction.

To get rid of the magnetic disturbance, we had to replace all iron and steel objects with a non-magnetic material. In our case, the best material to use is a wooden desk. As shown in Figure 4.8, we purchased a wooden desk specifically for this purpose.

We performed testing to see the effect of the avatar without magnetic objects nearby. The position of the avatar moving in a straight horizontal line was recorded at 5 cm intervals and composed by using Photoshop editor (Figure 4.9). As predicted, the simulation got much better and it behaved more like the real world.

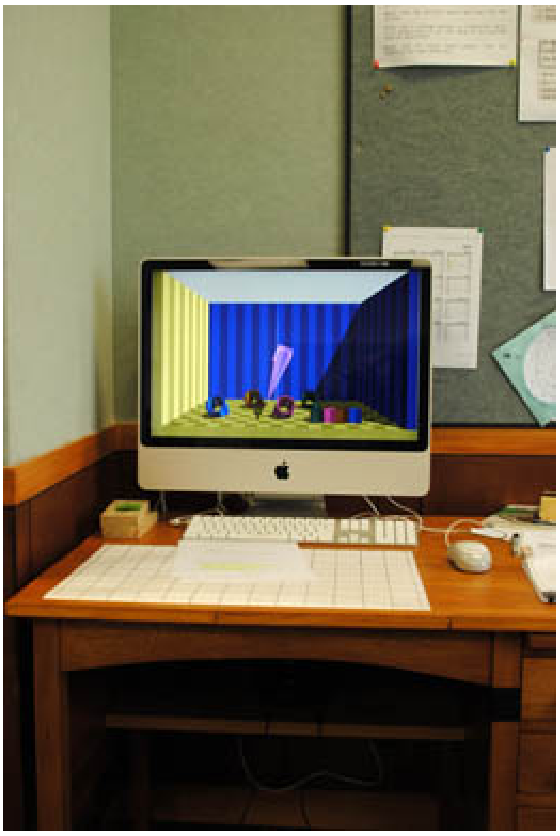


Figure 4.8: The setup of the hardware.

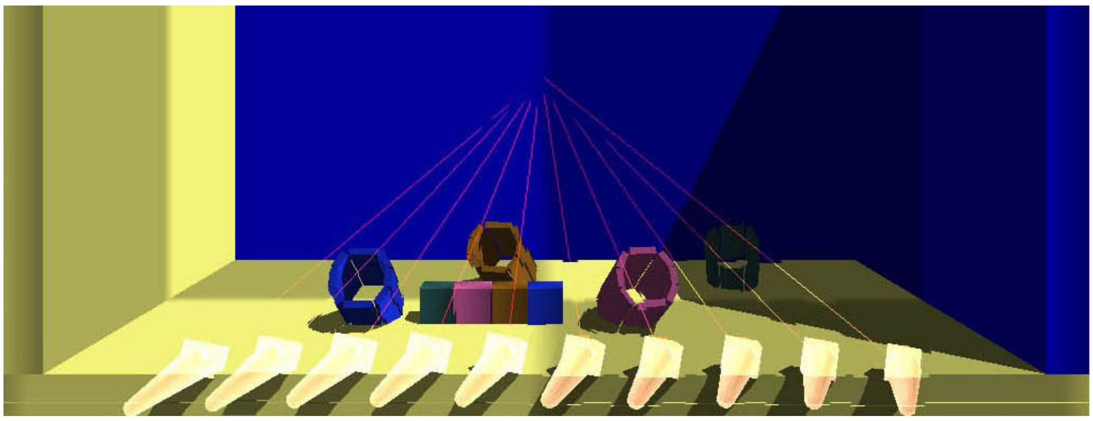


Figure 4.9: The horizontal movement of the avatar without iron or metal objects nearby.

4.2.2.2 Finding The Dependency Between Position and Orientation

We taped the controller onto a block of foam plastic which was placed on top of a square CD case as shown in Figure 4.10. This was to make sure the controller didn't move while the experiment was conducted. Then we moved the CD cover in a straight line and observed the movement of the simulated avatar. The orientation of the avatar didn't change when we displaced its position. The experiment was repeated several times and the result was the same for each. Therefore we can safely conclude that the orientation data returned by the controller is independent of the position.

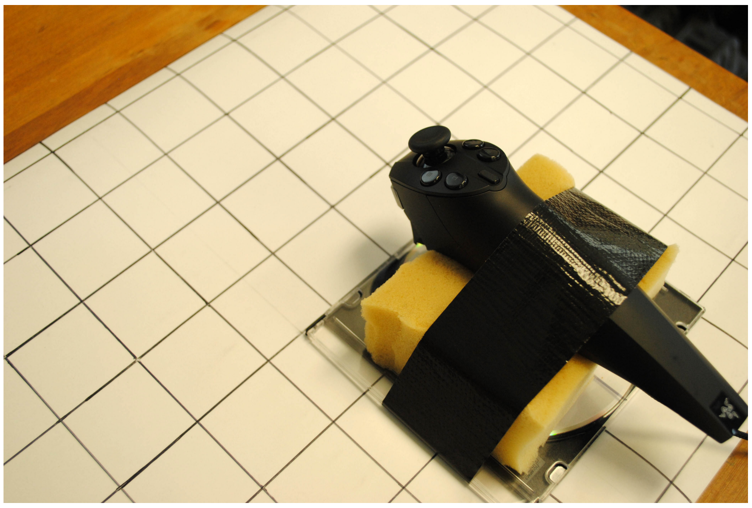


Figure 4.10: The controller is taped onto a block of foam on top of a square CD case, to test the orientation of the Hydra.

4.2.2.3 Finding The Position of The Base Station

The next step was to find the best position to place the base station. We did more tests to determine just that. The idea was to retrieve a set of points rotated in a circle and draw the points. Since we were dealing with 3D, the produced shape should be an ellipse rather than a circle. Then, we looked for the least distorted ellipse to place the base station.

Figure 4.11 shows the measurement device created for this particular experiment. The cylindrical spindle and cover from a CD package was used to make a simple rotating holder for the controller. The controller was attached to the disc cover and rotated where its position was recorded at every 30^o.

We got almost the same result when placing the base station anywhere on the desk. The best looking ellipses were produced when the base station was placed right underneath the desk. This result can be seen in Figure 4.13.

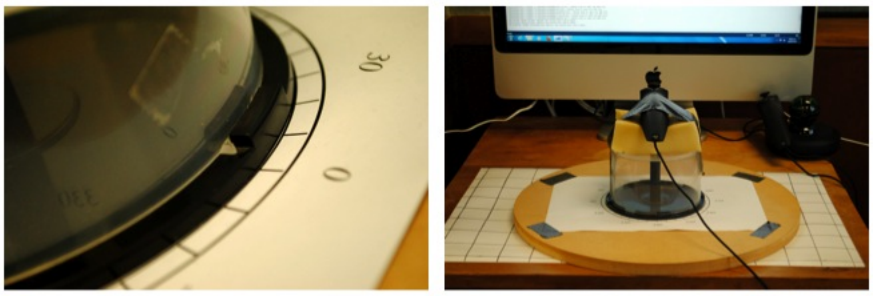


Figure 4.11: The device used to measure the rotation of the controller.

Figure 4.12 is an example of ellipses produced when the base station was placed inside a drawer on the right side of the calibration desk. From the figure we could say that the further the controller is from the base station, the more distorted the shape of the ellipse became.

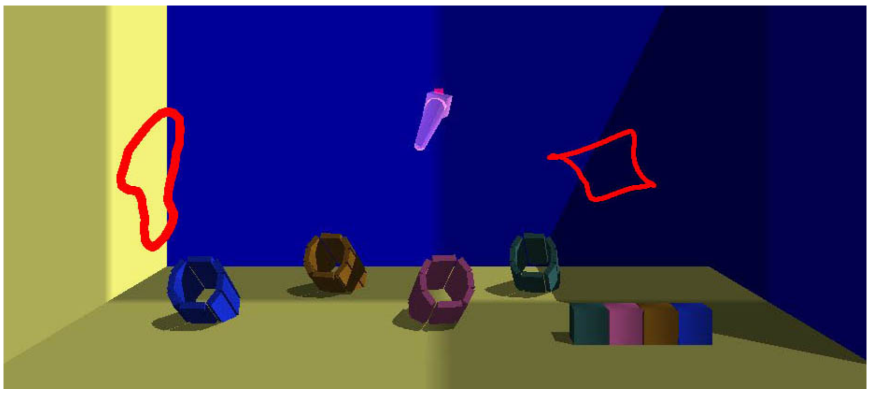


Figure 4.12: The ellipses produced when the base station is placed in the desk's right drawer.

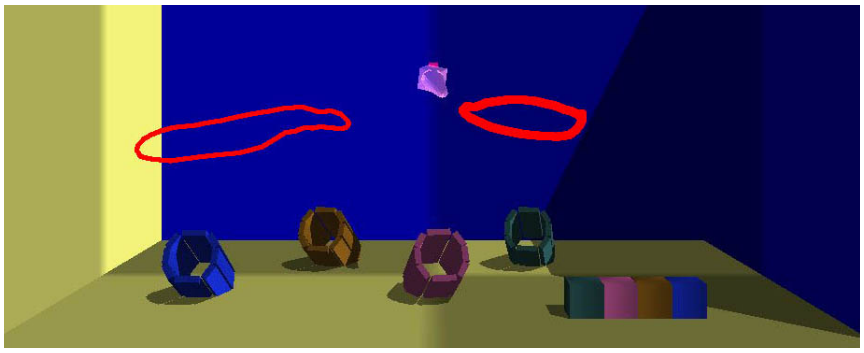


Figure 4.13: The ellipses produced when the base station is placed at the bottom of the desk.

Based on this little experiment, we created a new shelf to hold the base station permanently underneath the desk as in Figure 4.14.

4.2.3 Designing the Direct Manipulation Technique

Since our objective is to achieve naturalism through one-to-one correspondence, it is very important to make sure that the movement of the controller is mapped directly to the avatar. However, the coordinates of the controller are non-linear and there's no way to tell exactly where the coordinates are in space. Therefore these coordinates need to be calibrated in order to make the avatar move only within the screen display.

Before we proceeded with the calibration process, we needed to know the level of non-linearity returned by the controller. As usual, we tested this by retrieving the values from the controller at certain positions on the calibration sheet and plotted

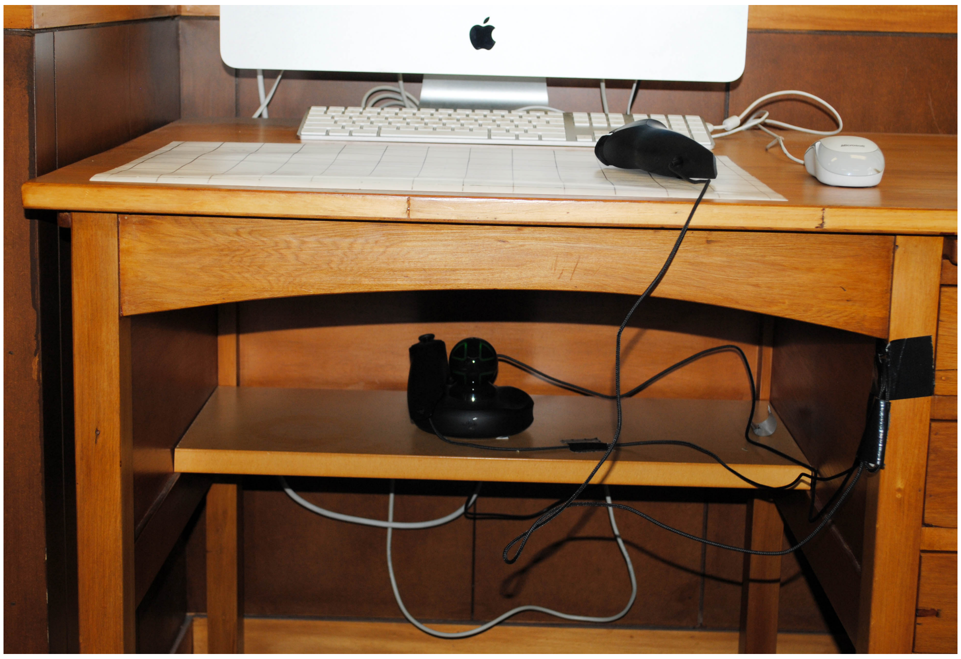


Figure 4.14: The base station is permanently placed on a newly created shelf underneath the desk.

them on a 3D scene to see what type of shape would be produced. From there, we determined what calibration method to use. This process is described in detail in Chapter 5.

4.2.3.1 The Interaction

The main interaction in this system is by using magnetic force to manipulate the 3D object. The magnetic interaction required is a simple attractive force. A simulation of a more complicated magnetic effect is out of the scope of this project.

To calculate the magnetic force, we needed to know the distance between two rigid bodies and how strong the force is. This calculation can be represented as:

$$f = \frac{k}{dist^2} \tag{4.1}$$

where k is some constant and $dist$ is the distance between two rigid bodies.

Our k value has to produce a force greater than the gravity in order for the cubes to be lifted from the ground. For this purpose, we simply chose k to be three times greater than the gravity. The reason was simple. First of all, all objects have the same mass, hence only one constant k value is enough for the whole interaction. The force applied to the object needs to be able to attract the object and at the same time it must be easy to release it. If the force is too strong, the naturalness of the interaction would be less because users would find themselves struggling to release the object from the magnetic grip. There is thus a tradeoff to be optimized in selecting the value of k. After experimenting with different values of k, we determined that the tradeoff is optimized when k is three times greater than gravity.

The magnetic point is somewhere inside the head of the avatar so we needed to take that into account as well. Therefore, the k value can be calculated as:

$$k = 3 \times gravity \times av^2 \tag{4.2}$$

where av represents the half size of the avatar's head.

The force is applied to the cubes if they are close enough to the avatar. The way to interact with the magnet is the same as in the real world. Users just have to pick up the cube using the avatar, move it to the correct tube and manipulate the cube so it fits perfectly into the tube.

The tube is designed to have a hole at the bottom of it. So if users pushed the cube too hard into the tube, the cube would be pushed out of it. The cube can also be pulled out from the tube due to the magnetic attraction. Users really need to grasp the idea of how the magnet works in order to release the cube from the attraction. The easiest way is to slide the avatar to the side of the tube once the cube has been placed inside it.

The pulling force works really well. However, when the avatar and the cubes collided, the collision didn't go into a rest state. The cubes kept on trying to penetrate the avatar and this caused jittering. This problem occurred because we scaled up the gravity vector earlier as described in Section 4.2.1, hence the simulation did not match up with the internal simulation step anymore.

To fix this problem, we needed to increase the time step frequency. Instead of calling the time step $\frac{1}{60}Hz$ for each frame, we called it 10 times. As expected, the simulation is now capable of handling the gravity and collision properly. With this, we have a complete working virtual world for manipulating 3D objects using the Hydra controller.

4.3 Programming the Interaction

Since we were incorporating the Bullet engine, we decided that it would be wise to use a working application as a template. Bullet provides several demo applications within its executable file. Because our purpose was only to get properly working physics, we opted to use the BasicDemo application as a template.

The programming tasks are listed below:

1. Designing the scene

   The original demo application consists of hundreds of small cubes falling down on a floor. We recreated the scene by drawing three walls, a floor, four cubes, four tubes and an avatar representing the Hydra controller. All of these objects were drawn by using OpenGL.

2. Linking the Hydra SDK with the application

   Once the avatar had been created, it had to be linked to the Hydra controller. The Hydra SDK returns the position and orientation data of the controller. Our task was to pass these data to the avatar object and apply Bullet's transformation function to manipulate the movement of the avatar.

3. Moving the center of rotation (CoR) of the avatar

   By default, the CoR of an object is drawn at its middle position. In the case of our avatar, the object is a compound shape, which means it was created by combining two primitive objects, a cube and a cone. Since we defined the primitive cube as the parent shape for the avatar, the default CoR was set to the center of the cube.

   After finding the estimated CoR as discussed in Section 5.1.1, we moved the current CoR to where it is supposed to be by creating a new compound object as a parent object and the CoR as its origin. The avatar was then added to this new compound shape as its child.

4. The magnetic interaction

   With the avatar fully functioning, we need to apply the magnetic force to it. First, we determined which point in the avatar should act as the magnetic point. Next, we defined a function that will activate the magnetic force. This function is called each time the application goes through every frame by using a step simulation function.

The algorithm for implementing the magnetic interaction is as follows:

```
for each cube object do
    for each face of the cube do
        Calculate its tracking point;
        Find which tracking point is the nearest to the avatar;
        if its distance from the avatar's magnetic point is within the defined
        magnetic range then
            Calculate the magnetic force between these two objects;
            Apply the force to the object;
        end
    end
end
```

Algorithm 1: Implementing the magnetic interaction.

5. The calibration

   We dedicated one full chapter just for calibrating the controller. The code written for the calibration is described in Chapter 5.

6. Completion time function

   To avoid inconsistency in computing the completion time for each participant, we automated the timing calculation. We set the start of the timer to be at the moment when the magnet is activated for the first time. The time for each cube is computed as follows:

```
for each cube do
    if the cube is placed fully inside the tube then
        Get the time for that particular cube;
        Set the magnet flag to true;
    else
        Clear the timer for that cube;
        Set the magnet flag to false;
    end
end
```

Algorithm 2: Computing the completion time.

4.4 Summary

Before implementing the interface, we identified what the requirements were. From there, we fixed the problems related to the Hydra controller as discussed in Section 4.2.2. The next step was to code the proposed design. In order to do so, we listed the programming tasks essential to this project.

As described in Section 4.3, we created the scene suitable for the task, linked the controller to our application, wrote the calibration program and the magnetic interaction and finally computed the completion time for the designed task.

In the next chapter, we describe the implementation of our calibration process in detail.

Chapter 5

Calibration of The Hydra Controller

Whilst the orientation data of the controller is quite accurate and needed no further correction, the position data on the other hand needed to be corrected. The returned data are non-linear. It is quite hard to tell where the points are in space. To solve this problem, the points need to be calibrated so that a one-to-one mapping can be created from the raw coordinates to the real world position.

There are two kinds of error produced by this type of hardware - static and dynamic errors. Dynamic errors are caused by the changing external electromagnetic fields and they change over time. Static errors are produced by the electromagnetic field distortions from the surrounding metal and they remain constant over time. In our case, we are dealing with static errors. These can be corrected through the calibration process.

5.1 Preliminary Calibration Process

5.1.1 Finding The Center of Rotation of The Controller

Rotating the controller shifted the position returned and thus changed the position of the avatar. The reason is the center of rotation of the avatar is different from the actual controller.

By default, Bullet sets the center of rotation of its object to be at the origin of the object itself. Our avatar was created as a compound object with the cube as the first child. Hence the center of rotation of the avatar is set to be at its head.

That is why when the controller was rotated, the avatar shifted by a certain amount. It rotated around the axis at the origin of its head, not around the actual axis of rotation which can be anywhere inside or near the controller.

Determining the center of rotation was a little bit tricky. There is not enough information supplied by the manufacturer. I went from one forum to another searching for information. From a discussion in one of the forums, it was revealed that the magnetic coil is built at the tail of the Hydra so the best guess would be for the center of rotation to be somewhere near this area.

To prove the claim, we did more tests. We used the same measurement device as in Figure 4.11 to determine which position produced the smallest ellipse. The only difference in this experiment was the way we place the controller on the disc cover.

As can be seen from Figure 5.1, there are a few dots drawn on the sponge. The dots represent the position of the controller. By using the wire that sticks out from the controller's tail as a marker, we placed the wire on one dot and recorded the points at every 30^oangle. This process was repeated for the other dots.

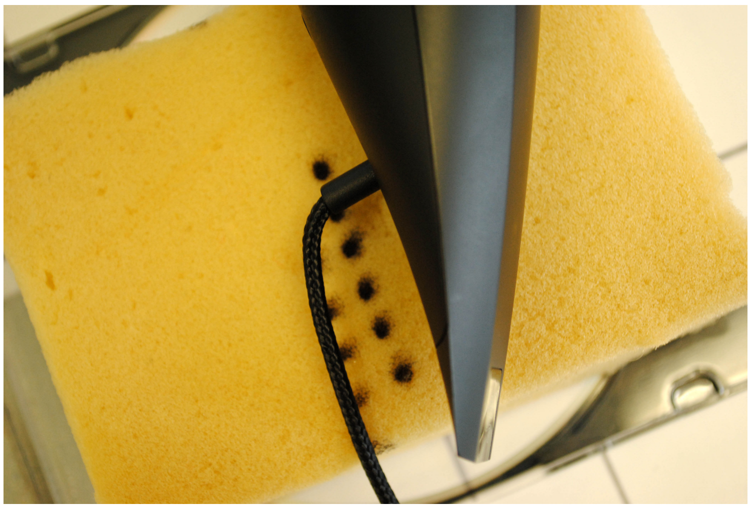


Figure 5.1: To find the center of rotation, the wire that sticks out from the controller's tail is placed on one of the dots to mark the possible position of its center.

Figure 5.2 shows the ellipses produced based on four dots position. We calculated the rotational position starting from the the third dot from the bottom of the sponge. The last two dots produced large ellipses so they were ignored for this purpose.

From the result, only the fourth and the fifth dots produced smaller ellipses. If we place the controller below the third dot or above the sixth dot, the ellipse would just get bigger. This test was repeated a few times to make sure we had consistency in the result. Hence we can conclude that the center of rotation for the Hydra is somewhere in between the fourth and the fifth dots. Based on that, we moved the center of rotation of the avatar to where we think it is supposed to be.

The final position of the center of rotation is shown in Figure 5.3. This time, when we rotated the controller, the avatar rotated accordingly.

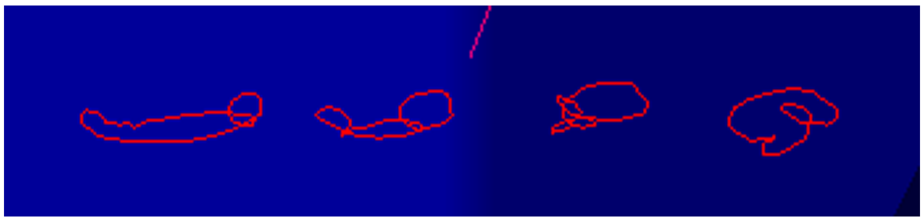


Figure 5.2: The shape of ellipses produced based on four different positions of the controller.

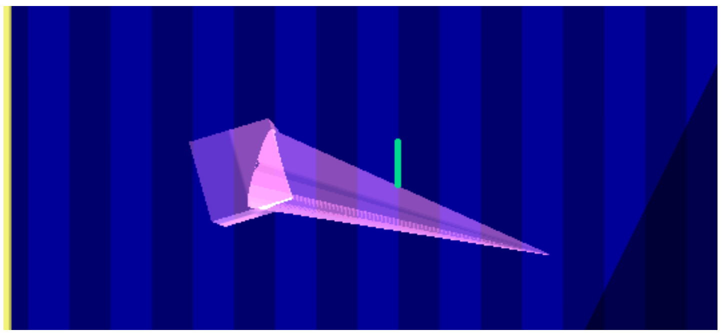


Figure 5.3: The center of rotation of the avatar is mapped to the correct position.

5.1.2 Determining the Accuracy of the Orientation Data

The orientation data returned by the Hydra controller seems to be accurate,. However, we still performed an experiment to determine the accuracy of these data and to measure any error produced by the controller. By using the same measuring device as shown in Figure 4.11, we adjusted the position of the controller taped on the disc cover by hand to get the value of the y-axis as close as possible to the vertical axis, although it is also possible to do the rotation around any arbitrary axis. There was no particular reason for choosing that way as the initial position, it is only to see how the vertical axis of the Hydra controller looks compared to the Bullet/OpenGL vertical axis.

We retrieved the value of Hydra matrices for every 10^o degrees, starting from the 0^o mark on the measuring device. For complete values of the rotation matrices, refer to Appendix E - Experiment. From the rotation matrices, we calculated a 'corresponding point' by multiplying each value with a unit vector and plotted the result. A circle was produced as shown in Figure 5.5. The circle is viewed edge-on in Figure 5.4 and you can see that it does not really lie in a plane. So a rotation that is strictly planar produces matrices from the Hydra that show a varying axis of rotation.

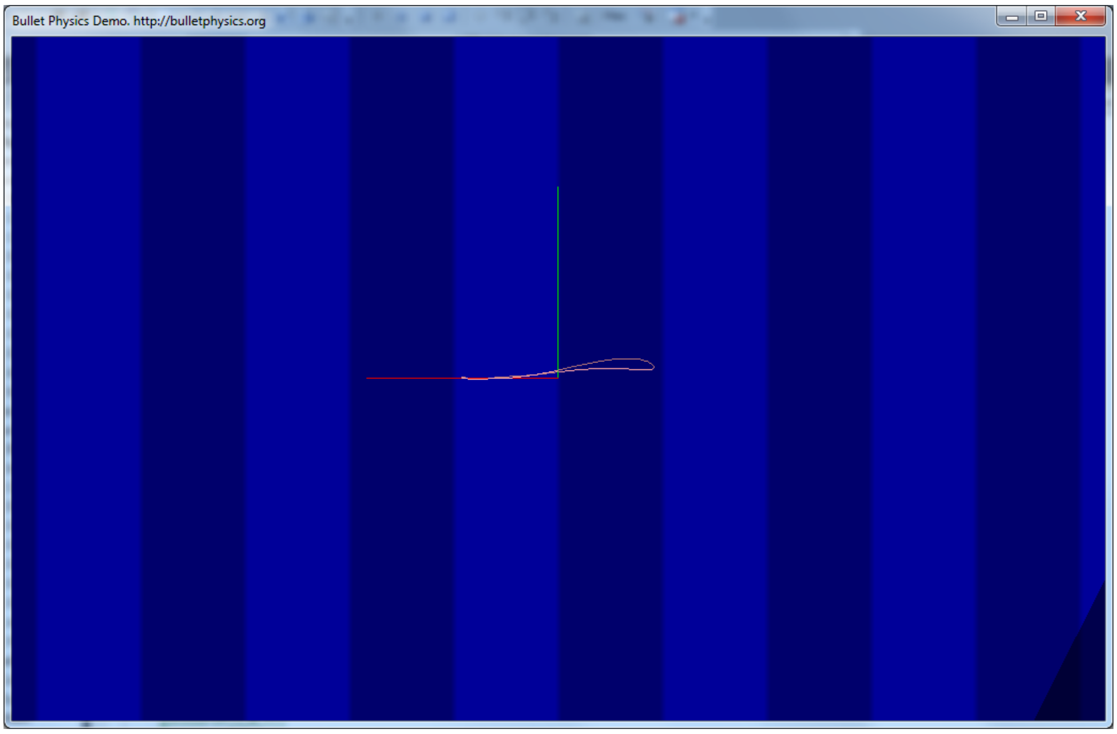


Figure 5.4: The plotted circle from the edge-on view

The next step was to compute the angle between each pair of points. This can be achieved by first, finding the center point of the circle, and then compute the angles. For accuracy, we calculated the angle for every 10^o using both dot product and cross product and recorded its value as shown in Table 5.1. The angle between two radii of the circle can be found from:

$$cos\theta = \frac{a \cdot b}{|a||b|} \text{ and } sin\theta = \frac{|a \times b|}{|a||b|},$$

where a and b are vectors generated from the center of the circle to each pair of points.

From the figure, the maximum quoted error is around $\pm 2^o$ which is good enough for our purpose. The manufacturer claimed of 1^o accuracy may still hold true. The controller motion and angle detection is sensitive to very small changes and it is very stable. So a tiny movement or rotation produces a convincing response in the avatar. The reported error is what we measured in our lab, and this can be caused by many factors. The lab is in a reinforced concrete building. Although we used a wooden desk and put all iron and steel objects away, there is still iron in walls and floor. Furthermore, the computer on the wooden desk may also influence the magnetic field. Intuitively, the avatar appears to follow the controller very well. The users seem to correct for the 2^o error naturally while manipulating the object.

Degress	**Angle(a.b)**	**Error produced**	**Angle(axb)**	**Error produced**
0-10	11.46	1.46	11.38	1.38
10-20	10.66	0.66	10.60	0.60
20-30	11.01	1.01	10.94	0.94
30-40	9.86	-0.14	9.81	-0.19
40-50	9.79	-0.21	9.74	-0.26
50-60	9.65	-0.35	9.60	-0.40
60-70	10.15	0.15	10.10	0.10
70-80	10.22	0.22	10.16	0.16
80-90	9.40	-0.60	9.36	-0.64
90-100	10.62	0.62	10.56	0.56
100-110	9.26	-0.74	9.22	-0.78
110-120	9.73	-0.27	9.68	-0.32
120-130	9.29	-0.71	9.25	-0.75
130-140	8.69	-1.31	8.66	-1.34
140-150	9.28	-0.72	9.23	-0.77
150-160	8.92	-1.08	8.88	-1.12
160-170	9.22	-0.78	9.18	-0.82
170-180	9.49	-0.51	9.44	-0.56
180-190	10.53	0.53	10.47	0.47
190-200	9.53	-0.47	9.48	-0.52
200-210	11.36	1.36	11.28	1.28
210-220	10.21	0.21	10.15	0.15

Continued on next page

Table 5.1 – continued from previous page

Degrees	Angle(a.b)	Error produced	Angle(axb)	Error produced
220-230	9.84	-0.16	9.79	-0.21
230-240	10.27	0.27	10.22	0.22
240-250	8.86	-1.14	8.82	-1.18
250-260	9.42	-0.58	9.37	-0.63
260-270	8.99	-1.01	8.95	-1.05
270-280	9.34	-0.66	9.30	-0.70
280-290	9.25	-0.75	9.21	-0.79
290-300	9.00	-1.00	8.96	-1.04
300-310	10.68	0.68	10.62	0.62
310-320	11.28	1.28	11.21	1.21
320-330	10.77	0.77	10.70	0.70
330-340	12.13	2.13	12.04	2.04
340-350	12.01	2.01	11.92	1.92
350-360	11.39	1.39	11.31	1.31
Maximum error (abs):			2.13	2.04

Table 5.1: The angle computed for every 10^o

We also determined the Hydra's vertical axis by computing a cross product between two vectors. Figure 5.5 shows the plotted circle and the axes. The red, green and white lines are the axes drawn based on the Bullet/OpenGL coordinates. The purple line denotes the Hydra axis calculated from the cross product of two vectors, which is not too far off the vertical axis in our scene.

From this experiment, we found that even if we try to fix the error, the Hydra continues to give periodic error. A maximum error at about 2^o is good enough for our purpose. Furthermore, based of the subjective measurement, the simulation of the avatar's orientation seems intuitive and behaves as desired, which meets the requirement of our natural interface.

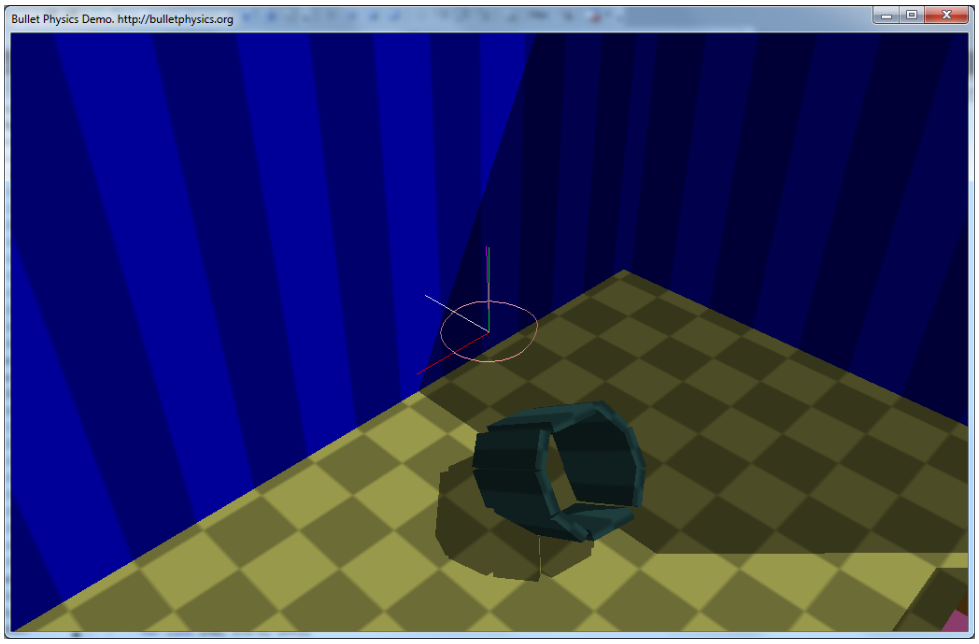


Figure 5.5: The Hydra rotational points and its axis.

5.2 Calibration Data Acquisition

Before implementing the calibration process, we must have knowledge of the shape of the grid we are trying to visualize. This shape must be in the form of a rectilinear grid. To obtain this, we drew a 52x30cm calibration sheet representing the $x-$ and $z-$plane in the real world space [Figure 5.6]. This measurement is chosen based on the physical size of the display. The height in $y-$plane is set to be 23cm, to extend the grid into 3D.

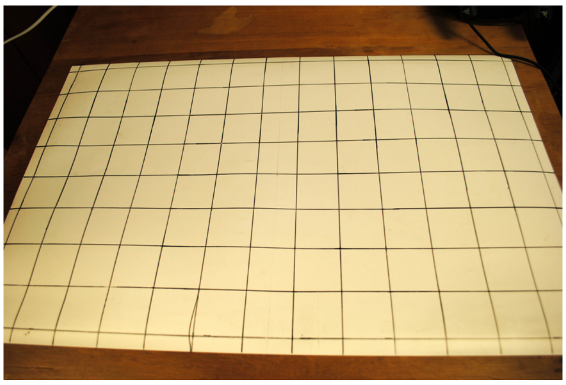


Figure 5.6: A 52x30 cm calibration sheet.

In order to have a rough idea of the position of the controller in the raw space, we performed a simple test. The rectilinear 3D grid is divided into two layers vertically. Each layer is further subdivided equally into four smaller sub boxes.

The raw coordinates at each corner of the sub boxes are recorded and drawn on the screen as shown in Figure 5.7. From the drawing, it can be seen that the shape produced is irregular. Based on the shape, we found it possible to perform correction on the raw coordinates using a simple calibration method. Video 1 - Rough Shape of the Working Space in Appendix B shows how these points were captured with the Razer Hydra.

The calibration process involves two sets of points in different spaces. The first set consists of raw coordinates read from the Hydra controller. The second set is the target coordinates, where the simulation of the 3D manipulation task will take place.

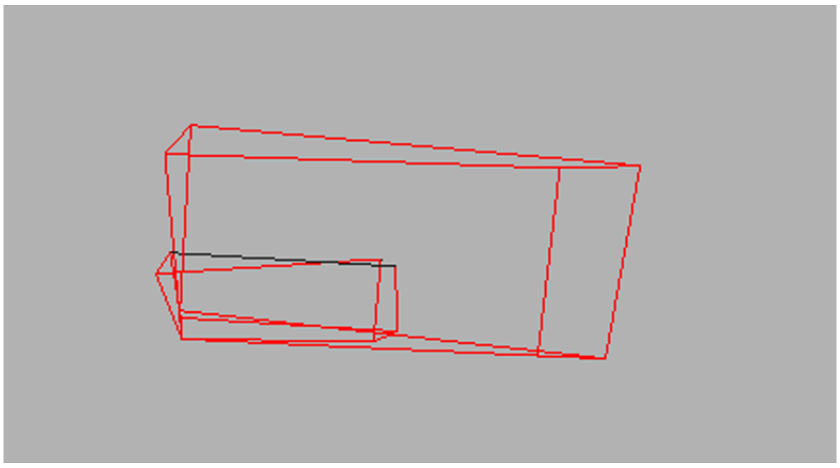


Figure 5.7: The rectilinear 3D grid for a sub-box and a whole box based on coordinates returned by the Hydra controller.

5.3 Position Data Correction Method

In a survey conducted by V. Kindratenko [62], there are two types of correction described; global methods and local methods.

5.3.1 Local Methods

These methods work by taking only some points to get a localized solution. One of the examples is tri-linear interpolation used by M. Ghazisaedy et al. [36] and M. A.

Livingston and A. State[71]. This technique uses the concept of barycentric mapping where the entire raw rectilinear grid is divided into smaller subboxes. Any raw point, P inside this subbox will be mapped to a point inside an arbitrary polyhedron in the target space.

Another method is called the tetrahedral shape function [118]. This works as an inverse of tri-linear interpolation. It takes an arbitrary hexahedron in the raw space and maps it into a corresponding cube in the target space. This method was not actually implemented. Its author just provides a theoretical framework for this concept.

The other local interpolation schemes used by S. Bryson [18], W. Briggs [16] and V. Kindratenko and A. Bennett [61] were done by calculating the localized interpolation from surrounding points. For each point in the raw and target space, its error vector is computed. Then for the point inside the polyhedron of the raw space, its weight is calculated from all eight corners.

These three methods require the raw space to be divided into smaller subspaces which means the quality of the calibration is dependent on the size of the subspace. The bigger the size, the lower the quality it might produce. The smaller the size, the more calculation it has to perform.

5.3.2 Global Method

There are a few correction methods implemented under this category. One of them is based on the high-order polynomial fit as demonstrated by S. Bryson [18] and M. Ikits et al. [53]. Given a rectilinear 3D grid in both raw and target spaces, an error vector is calculated for each point in the raw space. A degree r vector polynomial that fits the error vector will be computed and applied for correction.

G. Zachmann [117, 118] implemented a global error correction technique based on Hardy's Multi-Quadric scattered data interpolation method. It uses an interpolation function to correct the error. There is also a method which is based on a neural network. However, this implementation is not of our interest and will not be described in here.

5.4 Our Method

We implemented an approximate barycentric method that produced discontinuities. This problem happened because we were using hexahedral units for the mapping. We did consider implementing an exact barycentric method based on tetrahedra.

C. W. Borst [13] described a calibration method using a tetrahedral mesh. From the paper, this technique seemed to be quite complicated to code. It requires a lot of data points because for any point, P, the tetrahedron surrounding the point must be identified and a list of voxels containing the intersecting tetrahedron must be generated.

We prefer to use a simpler method. We decided that a very smooth fitting function would be better than one where the gradients could change discontinuosly at volume boundaries.

The alternative solution is to use a very simple rational weight function and then apply the error correction technique as in the local interpolation schemes. The idea is to use transformation matrices to give the best approximation of linear transformation and only then we compute the error vector for the points that can't be fitted linearly. Since the position and orientation data returned by the controller showed that the errors produced are not too big, implementing a local interpolation method seems like a good choice.

Initial informal experiments suggested that a linear fit to the whole space might work with a small non-linear correction. We tried with the implementation and this technique did produce a desired output. Hence we decided to proceed with it. The detail of this technique is described below.

5.4.1 Shear Transformation

Shear transformation preserves one plane. So we can use a sequence of shear transformations that preserve points that we have already fixed, to find a single matrix that makes a good approximate linear fit. Figure 5.8 shows the example of shear transformation in 2D. Given an irregular rectangle shape made up of four points, P_0, P_1, P_2 and P_3. We can fix three points in this example.

First, P_0 will be translated to the origin and fixed at that position. Next, P_1 is sheared in the $x-$direction and P_2 is sheared in the $y-$direction. Then we can apply the error correction method to fix P_3 which is in error.

5.4.2 Transformation Matrices

This theory can be easily extended to 3D. Extending one more dimension means one more point can be fixed at the intended position. The whole process can be achieved by multiplying each point, P, by a set of transformation matrices as below:

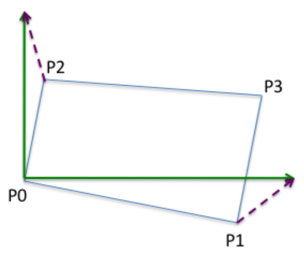


Figure 5.8: An example of shear transformation in 2D.

$$sH3 * sH2 * sH1 * T * P \tag{5.1}$$

T, $sH1$, $sH2$ and $sH3$ are four transformation matrices that will fix point P linearly. In 3D, there are eight corners in both raw space and target space. Each transformation takes place in different spaces. We consider the raw space as "space0". Hence the eight corners in space0 can be labelled as $P_{0:0}$, $P_{0:1}$, $P_{0:2}$, $P_{0:3}$, $P_{0:4}$, $P_{0:5}$, $P_{0:6}$ and $P_{0:7}$. The corners of the target space are S_0, S_1, S_2, S_3, S_4, S_5, S_6 and S_7. These are the known points in the real world sampled from the calibration sheet. We assume that S_0 is at the origin. This simplifies the construction of the shear matrices. S_0, S_1, S_2 and S_3 lie on the calibration sheet so $P_{0:0}$, $P_{0:1}$, $P_{0:2}$ and $P_{0:3}$ are measured directly with the Razer Hydra. S_4, S_5, S_6 and S_7 lie above the plane of the calibration sheet. These are measured by placing the Razer Hydra on a box as shown in Appendix B: Video 2 - Calibration Process.

5.4.2.1 The Translation Matrix, T

The first step is to shift $P_{0:0}$ to the origin of the target space, S_0, by multiplying it by the matrix T. Matrix T can be represented as the difference between $P_{0:0}$ and the origin:

$$T = \begin{bmatrix} 0 & 0 & 0 & -P_{0:0x} \\ 0 & 0 & 0 & -P_{0:0y} \\ 0 & 0 & 0 & -P_{0:0z} \end{bmatrix} \tag{5.2}$$

5.4.2.2 Shear Matrix 1, $sH1$

Since the first transformation has been performed, all the points, P are now in space1. The next step is to shear three more raw points to the target space. The order of which points being sheared first doesn't really matter. In our case, I opt to shear in the $x-$direction first.

To define matrix $sH1$, we calculate the error vector,

$$\underline{v} = S_1 - P_{1:1} \tag{5.3}$$

We find the values of the 'x components' in the matrix $sH1$ by dividing each component of $\underline{v}$ by $P_{1:1x}$.

$$sH1 = \begin{bmatrix} 1 + \frac{v_x}{P_{1:1x}} & 0 & 0 & 0 \\ \frac{v_y}{P_{1:1x}} & 1 & 0 & 0 \\ \frac{v_z}{P_{1:1x}} & 0 & 1 & 0 \end{bmatrix} \tag{5.4}$$

5.4.2.3 Shear Matrix 2, $sH2$

The same concept is applied to the second shear transformation. I define $sH2$ to perform the shearing in the $z-$direction. The error vector is

$$\underline{v} = S_2 - P_{2:2} \tag{5.5}$$

and the shear matrix, $sH2$ is

$$sH2 = \begin{bmatrix} 1 & 0 & \frac{v_x}{P_{2:2z}} & 0 \\ 0 & 1 & \frac{v_y}{P_{2:2z}} & 0 \\ 0 & 0 & 1 + \frac{v_z}{P_{2:2z}} & 0 \end{bmatrix} \tag{5.6}$$

5.4.2.4 Shear Matrix 3, $sH3$

The error vector is

$$\underline{v} = S_3 - P_{3:3} \tag{5.7}$$

and the shear matrix, $sH3$ is

$$sH3 = \begin{bmatrix} 1 & \frac{v_x}{P_{3:4y}} & 0 & 0 \\ 0 & 1 + \frac{v_y}{P_{3:4y}} & 0 & 0 \\ 0 & \frac{v_z}{P_{3:4y}} & 1 & 0 \end{bmatrix} \tag{5.8}$$

Figure 5.9 shows how the raw coordinates look relative to the target coordinates after they have undergone all four transformations.

5.4.3 Non-Linear Correction

Since there are no more points that can be transformed linearly, this process leaves the rest of the points, namely $P_{4:3}$, $P_{4:5}$, $P_{4:6}$ and $P_{4:7}$ in error. These errors can be corrected non-linearly. The idea is to add a weighted sum of the errors at each $P_{4:i}$ to every point in this space.

For each point, P, we generate a calibrated point, *cp*, with

$$cp = P + \frac{\sum w_i * v_{ci}}{\sum w_i} \tag{5.9}$$

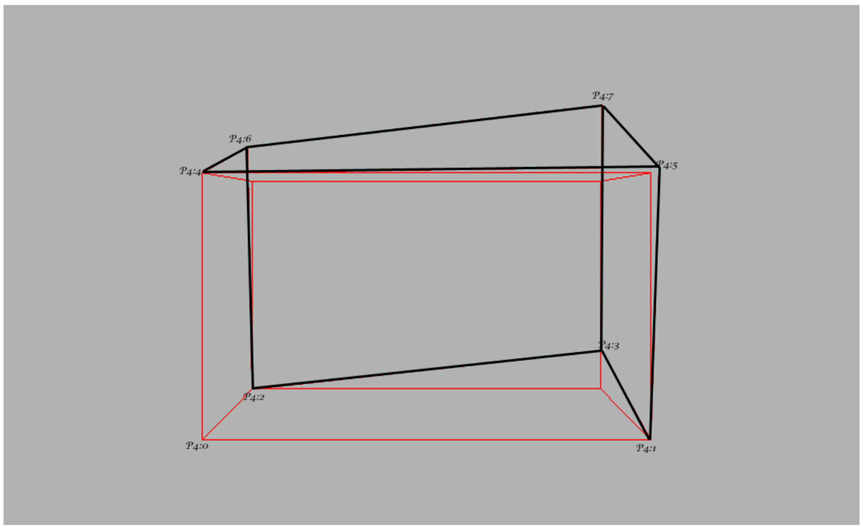


Figure 5.9: The rectilinear 3D grid after the final transformation. $P_{4:3}$, $P_{4:5}$, $P_{4:6}$ and $P_{4:7}$ are in error.

where i represents the number of each point from 0-7 and the errors, v_{ci} are

$$v_{ci} = S_i - P_{4:i} \tag{5.10}$$

The weights are based on a simple function of distance.

5.4.3.1 The Weighting Function

The next step is to calculate the weight, w of point P from all the corners. Since the errors in $P_{4:3}$, $P_{4:5}$, $P_{4:6}$ and $P_{4:7}$ are not that big, a simple function of distance, f is enough to give the right weight. We use

$$f = \frac{1}{0.0001 + d_i} \tag{5.11}$$

where d_i is the distance from P to $P_{4:i}$. The value of 0.0001 is added to the function to prevent a division by zero whenever d_i approaches a very small value.

5.4.3.2 The Final Transformation

This calibrated space needed only an additional shift to create the coordinates used by OpenGL/Bullet.

If Figure 5.9 shows the four positions that have been corrected linearly, Figure 5.10 shows the comparison between the Hydra returned position after the final calibration process (points in green) with the actual target points (in red). This means the green points represent the movement of the Hydra controller which is not mapped exactly to the actual target position. Since we performed linear correction on only four points, and corrected the other points in error non-linearly, it was expected to get a slightly off position at and around the middle area of the 3D grid.

For each pair of points, we computed its error distance. The maximum linear error produced was 6.19cm. For our purpose, this maximum value is acceptable. When manipulating the object, the users are looking at the avatar image on the screen, learning the avatar's behavior without needing to know the absolute reference for the Hydra controller, hence making the error unnoticeable. A complete list of the linear errors produced by the Hydra controller is provided in Appendix E - Experiment.

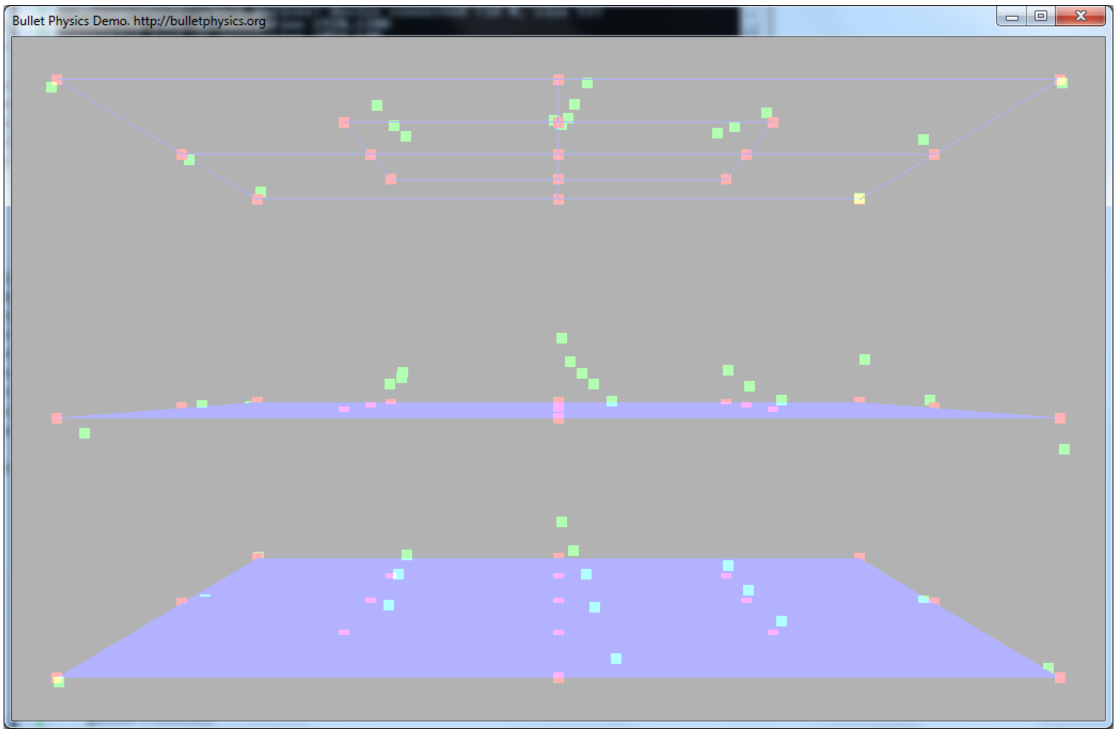


Figure 5.10: The returned Hydra positions vs the actual target position.

5.5 The Calibration Class

The calibration process is a big task, hence, I created a new class just for handling it. Since this process involves transformation of matrices, I also defined a 4x4 matrix class to take care of the maths.

The calibration class basically deals with shearing the matrices, calculating the error vector and applying the filter function to correct the error. Extra precaution has to be taken when transforming the matrices. As explained earlier in Section 5.4, each transformation matrix is done in its own space. I have to make sure that all the corner points were transformed to the correct space after each transformation, otherwise the controller will be wrongly calibrated.

We placed the Hydra base station at a fixed position which makes it easier to automate the calibration process each time we run the application.

The algorithm for calibrating the controller is as follows:

```
for every point in the raw coordinates do
    Apply the transformation matrices;
end
for a ppoint in space do
    Calculate its error vector from each corner;
    Calculate the filter function;
end
Apply the ratio of sum of all (errors*its weight) / sum of all weights;
Apply the ratio to the point in space
```

Algorithm 3: Calibrating the controller.

5.6 Summary

In this chapter, I have described a method for calibrating the position data obtained from the Hydra controller. This method is simple to implement and is less complicated than the high-order polynomial fit. Since we use the whole rectilinear 3D grid, there is no more problem of the avatar crossing boundaries. We apply the same concept used in the local interpolation schemes to compute the error vectors and the weights. The calibration proves to be really smooth and it maintains the one-to-one mapping concept that we want to achieve. The next chapter describes the testing of our natural interface.

Chapter 6

Testing The New 3D Interface

6.1 The plan

The user testing phase for this research compares two different interfaces - a standard interface that uses a mouse to perform a 3D manipulation task and the so called natural user interface which uses the Razer Hydra controller.

An important issue in our test of our new interface is deciding on a system to compare it against. One option would be to compare it to a custom-built interface exactly like our Hydra interface, but using a mouse-based system for 3D object manipulation, perhaps a system like N. Zhao's [120] mouse-based interface. The main reason for not using a custom-built mouse-based interface as a comparison is a methodological one: it would be rather easy for us to deliberately create a "bad" mouse-based interface to pit against our own proposed interface. By choosing a widely-used commercial mouse-based interface, we can keep ourselves honest: this is an interface which is known to work, and therefore provides a better comparison. As a separate decision, as discussed in Section 1.3, we decided not to tweak the Blender interface to configure it for the task at hand. The Blender interface is intended to be a general-purpose interface, and so we used it without special configuration.

6.2 The Task

The task conducted for this test must:

- be genuinely 3D and not a special case done easily with a sequence of 2D actions.
- be understandable in a few minutes and be simple enough to test in a few minutes.

- mimic a manipulation that would be easy in the real world.
- be complex enough to be representative of useful actions.
- not require human grasping.

For this purpose, I designed a 3D scene which consists of a set of cubes and tubes. The task is for the users to pick up a cube by using the Hydra controller and place the cube inside the tube that matches its color. While performing the task, the users are expected to manipulate the cube in 6DOF. The avatar behaves like a magnet as the means of interaction.

A similar scene has been developed in Blender, an open source application for 3D modeling. There were three reasons for choosing Blender to represent the mouse-based task. Firstly, it is free software, which can be obtained easily. Secondly, we had access to a reasonable number of Blender users. This was important because we did not have to provide any training for the participants before they took part in our experiment. Lastly the Blender interface works the same way as the other CAD software such as Maya and 3ds Max. The manipulation requires the participants to use different views and different modes for grabbing and rotating.

The scene in the Blender interface consists of a set of cubes and holes similar to the original scene, but the room itself is built without the walls as shown in Figure 6.1. This is because the walls will obstruct the scene when the view is changed from front to side and so on. For this interface, users are required to use the standard mouse to perform the task.

Manipulating an object in the Blender interface is similar to manipulating an object in a standard modeling interface. We did not add any extra interaction to it. The users have to select the cube, define the mode for grabbing or rotating, and then manipulate the object by changing the view of the scene until the cube is properly aligned inside its correspondence tube [Figure 6.2]. For a better understanding of the Blender interface, we provided a demo video which can be found in Appendix B: Video 4 - Blender Interface Demo.

6.3 Ethics

The experiments with human subjects were conducted according to the University of Otago's ethics policy and with all necessary approval. A copy of the form is attached in Appendix A.

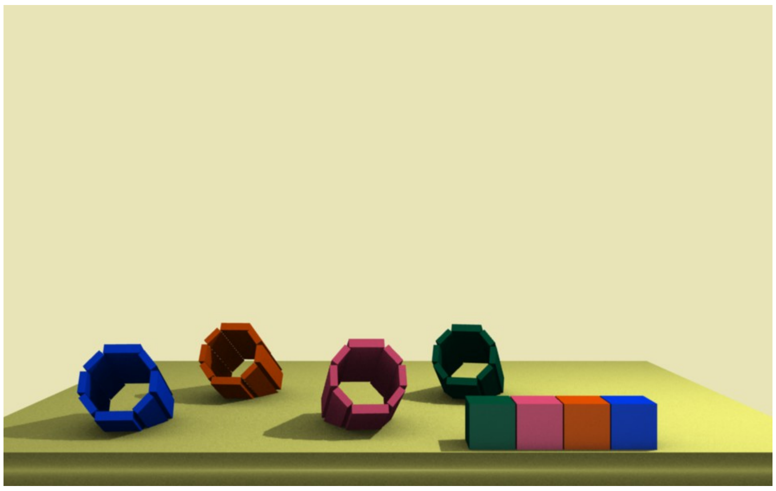


Figure 6.1: A rendered Blender scene.

6.4 The Participants

The aim of this experiment was to determine the ease of use of the interface. We did not ask for the users' background. They might be gamers or Hydra users. They might already have an experience in 3D modelling. This will give them an advantage in using our natural interface. However, we can guarantee that none of the participants have experience performing a 3D assembly task using magnetic force. Thus, in this context, we could say that the participants were using the system without prior knowledge.

The participants were selected based only on these two criteria:

1. Right-handed

2. Able to see the screen clearly

For the first criterion, we are not claiming that the left-handed people are not suitable. The main reason for just using right-handed people is that it is a standard procedure in experiments on motor skills in psychology. There are differences in motor control between right- and left-handed people[84]. The cubes are drawn on the right side of the scene. Left-handed people are most likely to take extra time to pick up the cubes since they have to move the controller from the left to the right each time. By considering only right-handed users, the evaluation will be fairer.

Since the task is in 3D and it also requires the participants to place each cube into a tube that matches its color, it is very important for them to be able to see the screen

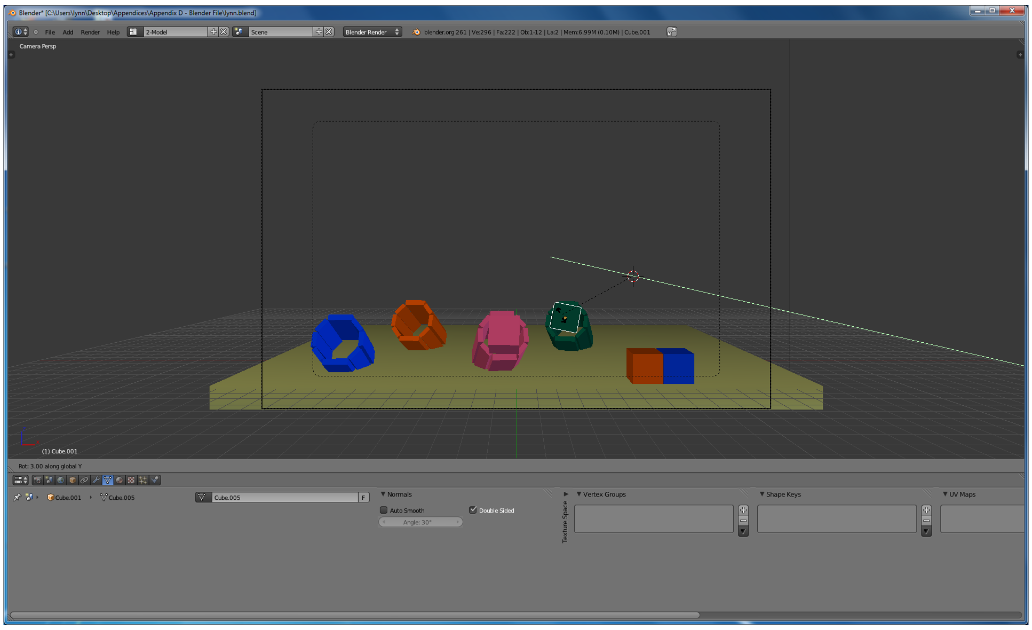


Figure 6.2: A screen shot from our Blender interface.

without any problem. By imposing these two criteria, we can maintain consistency throughout the process.

For the experiment on the Blender interface, there were five participants. Two of them can be categorized as expert users whilst the other three are considered to be average users. We define the expert users as those who have a lot of experience in using the Blender interface. The average users on the other hand were third year Computer Graphics students who had been taught to use Blender in an earlier course. They had typically used the software for six months to a year.

There are two decisions we need to justify about our choice of participants. Firstly, why did we not use the same group of participants for both interfaces? We could have certainly used a within-subject design for the experiment, comparing each user's performance on the two interfaces. Secondly, given that we used separate groups for the two interfaces, why did we use a group of experts for the Blender interface? Our reasoning is as follows. Firstly, we did not want to have to train large number of participants in the use of Blender. This would have taken far too long, and we did not have the resources. We only had a small group of Blender users, so the group using the Blender interface was constrained by practicalities to be small. At the same time, we wanted to test the new interface on a larger group of users. It thus made sense to use a between-subjects experiment, with two separate groups. Secondly, while it is clear that our comparison is skewed, because we are pitting a group of experienced users (of Blender) against a group of novices (for the new interface), this bias is in a good direction: if we can show the new interface is easier to use **even given no prior experience**, then we are providing very strong evidence that it is a better interface for the task.

6.5 The Test

6.5.1 The Natural Interface

Each participant was given a 15 minute slot. Before taking part in the experiment, the participants were presented with a short training video (Appendix B: Video 3 - Instruction for the Experiment) that shows how the task will be performed and how to use the controller. After watching the demonstration, the participants were allowed to perform the task as many times as they liked. For each attempt they made, their completion time was recorded. It was not compulsory for them to perform a second trial and there was no limit on how many times they could perform the task either.

The completion time was measured automatically by the application. The start time was determined the moment the cubes were being attracted by the magnet. This allows the participants to get the feel of using the controller before picking up the cubes. The time was also recorded for each cube that had been placed correctly inside its corresponding tube. The task was considered complete after all cubes had been put into all tubes. The completion time was calculated by taking the difference of the time of the last cube being put into the tube and the start time.

6.5.2 The Standard Interface

As explained in Section 6.2 above, the expert users of the Blender interface performed the same task as those taking part in the natural interface experiment. The only differences are the input used: the mouse and the way of manipulating the 3D object.

To start using the interface, users might want to switch the current viewing of the interface into a desired view or perspective. This can be done by pressing keys on the number pad (if using the standard keyboard): 0 will change the view to the default camera view as shown in Figure 6.2, 1 for the front view, 3 for the side view, 7 for the top view and 5 to togle between the orthographic and perspective views. To zoom in/out, the "+"/"-" key is used, or users can rotate the middle wheel of the mouse which does the same thing.

To select or deselect all objects, the "a" key is used. The right mouse button is used to select one object, or the user can hold the shift key together with the right mouse button to select multiple objects. This type of object selection can only happen in the "object" mode. The selected objects are outlined in an orange color (Figure 6.3[1]).

Once selected, the object can be translated across the scene. The "g" key puts the selected object in a translation mode (Figure 6.4). To translate the object along a specific axis, the "x", "y" or "z" key can be used. To see if the object is aligned correctly with the tube, users can change the viewing angle of the 3D scene by pressing the mouse wheel (middle) button (Figure 6.5). Similarly, the object can be rotated by pressing the "r" key (Figure 6.6). Pressing the "x", "y" or "z" key while the object is in the rotation mode will make the object rotate about the specified axis. In translation and rotation modes, the objects selected are outlined in white. The last step is to push the cube correctly into the tube. Even so, aligning the cube properly can require a bit more manipulation. Otherwise the cube might end up with its edge penetrating the tube as shown in Figure 6.7.

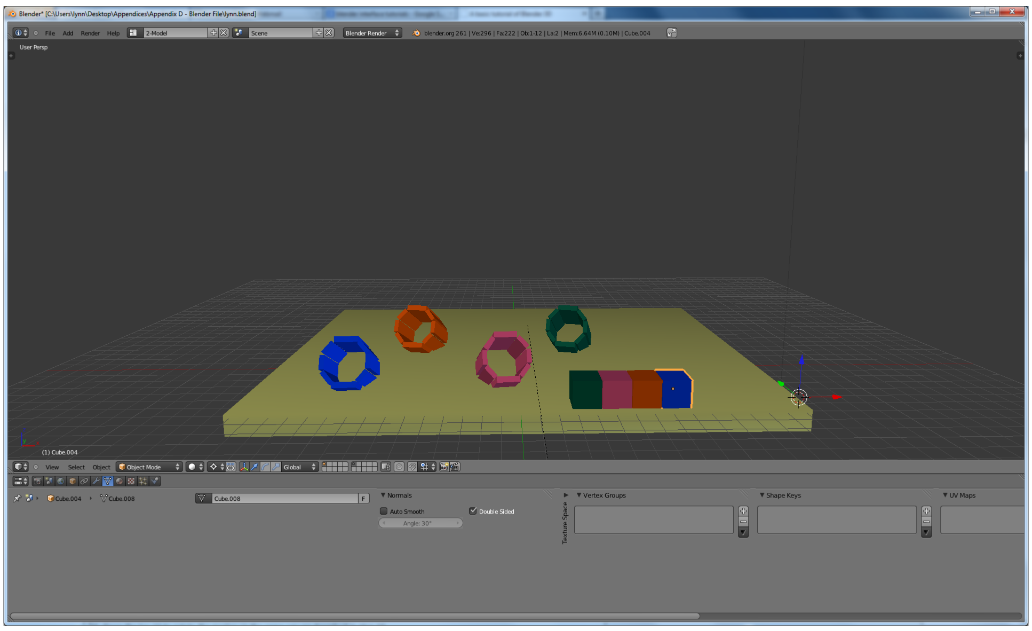


Figure 6.3: An object is selected.

[1]For a better visualization in the thesis, all outlines has been thickened by using Photoshop.

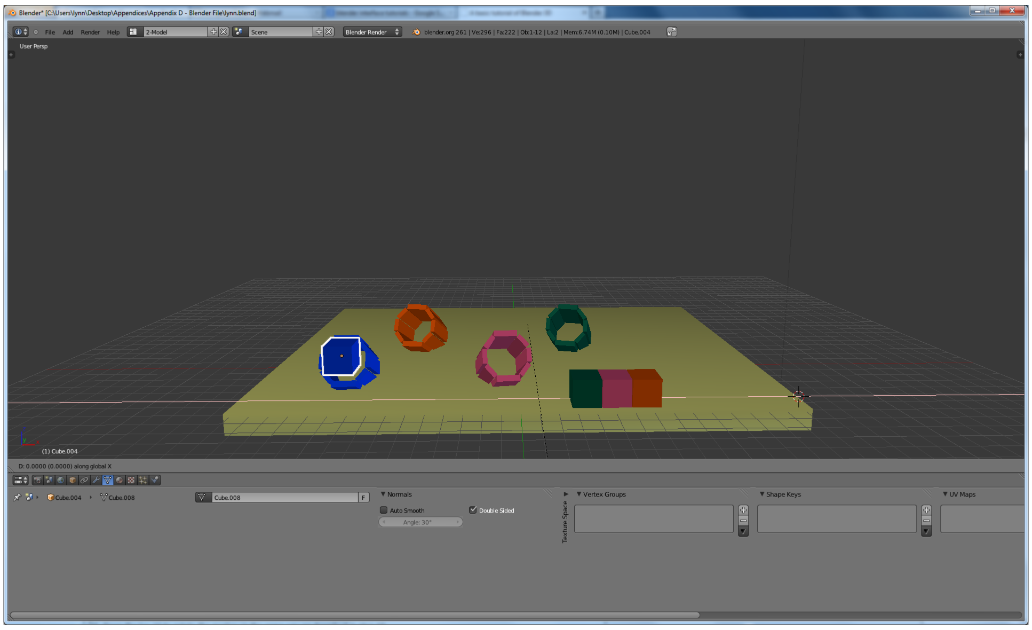


Figure 6.4: The object is translated across the scene.

6.6 The Evaluation

For experiment on both interfaces, the completion time of each participant was recorded. At the same time, their behavior while performing the natural user interface test was observed. We also asked the participants to rate the ease of use of the interface on a scale from easy to hard.

The completion time between the natural and the standard interfaces is compared. It was not practical to compare the interfaces under identical conditions. The natural interface was tested on subjects with minimal instructions, but the Blender interface is virtually unusable without detailed knowledge and we had access to only five skilled Blender users. So, while we were able to produce basic statistics for our main group, our comparison is simply made with the average completion time from the Blender group.

We performed basic statistics on the natural interface test because we wanted to know:

- How much better are the best people than the worst people?
- Are there any people who find it very difficult?
- Is everyone able to use it sucessfully?

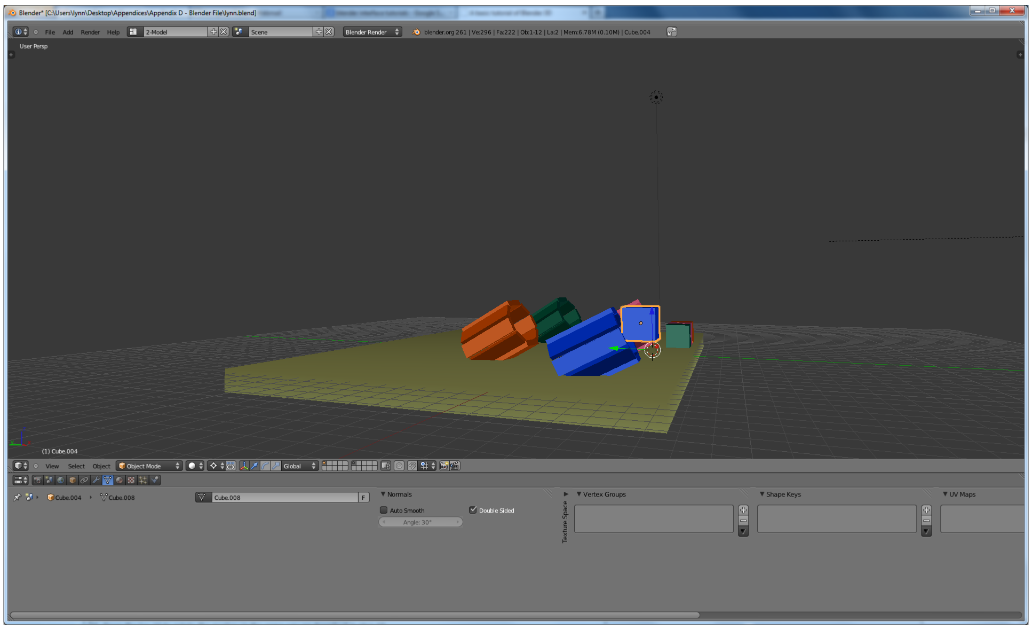


Figure 6.5: Changing the viewing angle of the 3D scene.

We used 19 student volunteers with a variety of backgrounds. Before the experiment took place, we performed a pilot test with several postgraduate students on the natural interface to make sure the actual experiment could be conducted without errors.

6.7 Methods Comparison

In this section, we summarize the differences between the natural and the Blender interfaces in terms of the task performed, the participants and the interaction techniques.

1. Task performed

   For both interfaces, the task remains the same, which is to place each cube into its corresponding tube.

2. Participants

   We used a different group of people for each interface. The reason was that novices would take days to learn to use the Blender interface adequately. We needed subjects who have already had the experience of using the Blender interface.

   The natural interface on the other hand, requires almost no instruction. We only specified that the users must be right-handed and are able to see the screen clearly to ensure fairness in the experiment.

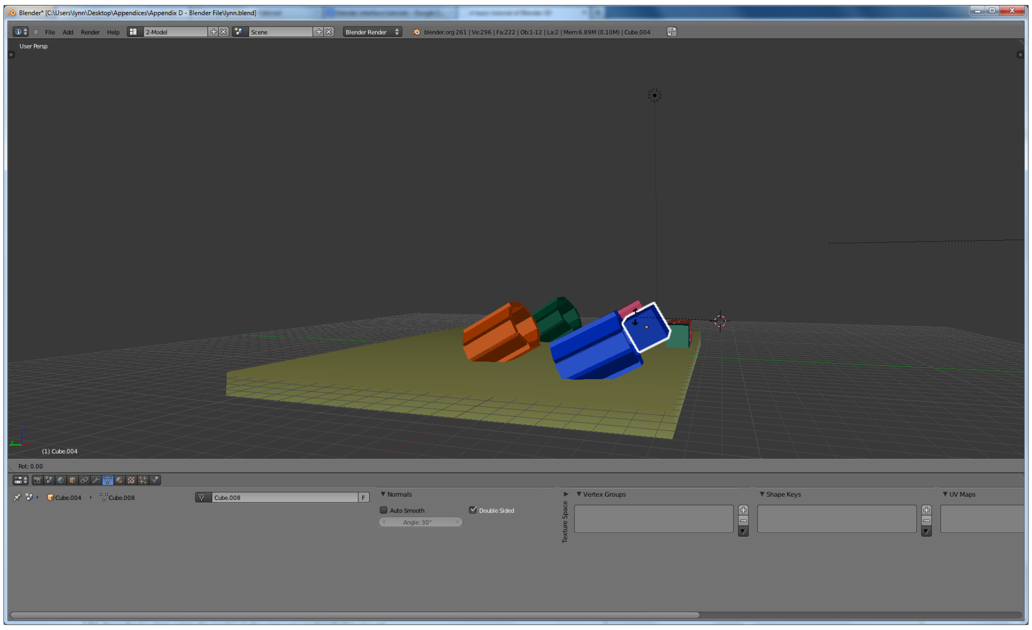


Figure 6.6: Rotating the selected object.

3. Interaction techniques

   For the natural interface, we used the Hydra controller as the input. The interaction was implemented by applying a magnetic force to the controller so that the cubes will be attracted to it. The manipulation were done in 6DoF, as if the users were performing the task in the real world.

   For the standard modeling software, the mouse was used. The 3D manipulation was done in different modes (for translation and rotation) and different views (for aligning the cube inside the tube).

   There is a huge difference between the natural interface and the standard interface in terms of the interaction techniques they employ. The NUI uses magnetic force to pick up the object whereas the mouse relies completely on the conventional way of selecting and dragging the cube. One might argue that a fairer experiment would employ a concept of magnetic force in the standard interface as well; however, this concept is a central part of the new natural interface which we want to evaluate; we explicitly want to compare an interface which uses this concept to a standard interface which does not.

We discuss the results of the experiment and the observations made in the next chapter.

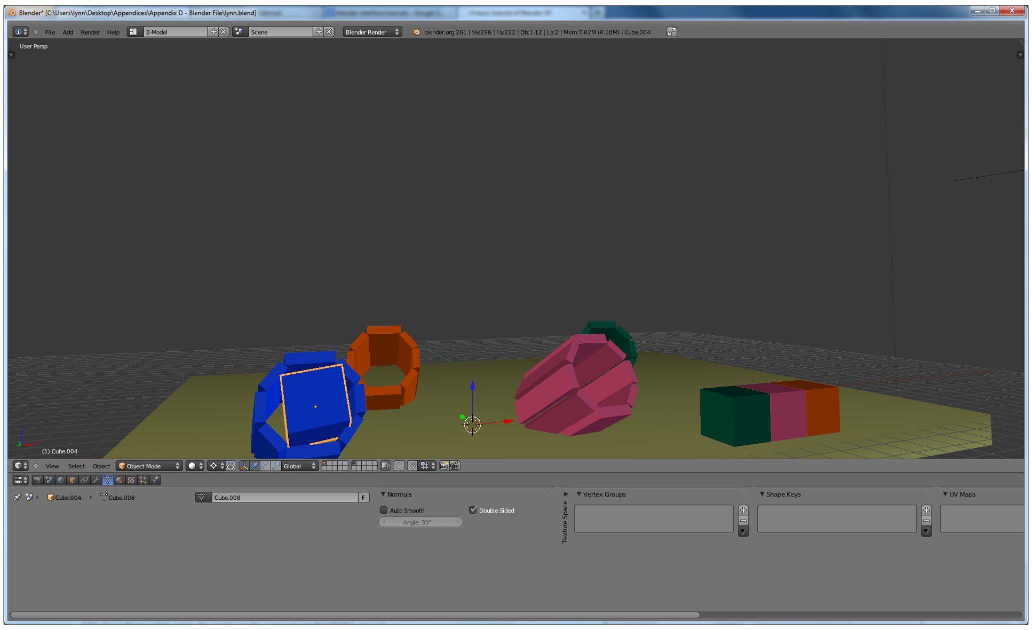


Figure 6.7: The cube is not properly aligned with the tube.

Chapter 7

Results and Discussion

Without giving too much instruction, we need to know how well the participants can use the natural interface. To maintain the consistency in giving out the instruction to the participants, we used only the training video to show them how to perform the task [Appendix B: Videos 3 - Instruction for the Experiment] . From the video, the participants have to figure out how to manipulate the 3D object in the actual test.

For the Blender interface test, we just asked the participants to place the cubes into the tubes. How to perform the task was completely up to them. We just wanted to observe how long they took to manipulate the 3D objects by using a standard method.

7.1 The Natural User Interface Test

19 students volunteered to take part in the natural interface test. As mentioned in Chapter 6, their background, gender and age is of no importance to our purpose because we are not testing on how these factors can affect their performance. We want to find out only if the natural interface is easy to use and how fast they could perform the task.

7.1.1 The Result

For the task performed, we calculated the standard deviation and the mean of the completion times. We need to determine the average time taken to complete the task, how many students performed within the standard deviation of the mean, and how many of them are much faster or slower.

7.1.1.1 First Attempt

Timing the first attempt is quite unusual in most situations but in our case, it is essential. Our definition of NUI states that the user should be able to use the interface with only almost no instructions and this can be found out only on the first attempt.

There were a few interesting observations made while the participants were performing the task. Many of them took a longer time to complete the task because they used their first trial to gain understanding on how the interface works since the only information they had was from watching the training video.

Whilst this is definitely the right way to learn the interface, most of them seem unable to understand that it was a 3D task. For this trial, the only information they had was from watching the training video. No additional instructions were given.

They understood the idea of using magnetic force to pick up the cube. However they were confused on how to release the magnetic grasp. Some of them saw straight away that they needed to slide the avatar to the side of the tube. Others tried to push the cube too hard into the tube which eventually pushed the cube out of the tube through the hole at the tube's end.

The transparency of the avatar, as well as its attribute as a kinematic object also contributed to the longer completion time. Most of them fail to treat the avatar as a 3D object. A kinematic object can go through any static object but will collide with dynamic objects.

Without understanding this concept, the participants simply dragged the avatar with a cube attached to it, to pass through the tube. This caused the cube to be released from the avatar due to its collision with the tube.

Another similar mistake was made after putting the cube into the tube. While moving the avatar, they simply moved it around without trying to avoid the tube. This action caused the cube to be knocked out of the tube and increased the completion time because they needed to put the cube back into the tube.

However, one major discovery revealed from the test is the inability of majority of the participants to manipulate the 3D object in 6DOF. Each tube is oriented slightly differently from the others. In order to place the cube correctly, the participants needed to control the avatar to align the cube properly with the tube, but that didn't happen much during the test. They simply try to place the cube without rotating the avatar, increasing the difficulty of completing the task.

Table 7.1 shows the completion time for the first attempt for each participant. The numbers on the left hand side of the table refer to the participants' number in order. Based on the performance, the calculated mean is 85 seconds and the standard deviation is 46 seconds.

7.1.1.2 Second Attempt

After completing the first trial, the participants were asked if they would like to try for a second trial. All of them decided to go for the second trial. However, one of the participants accidentally knocked his last cube out of the tube, hence his completion time is not recorded. Therefore, for this second attempt, we only use the result of 18 participants.

Table 7.2 shows the times recorded for the second attempt. The concept of this second trial is the same as the first. No additional instructions are given to them. Although there is a significant improvement in the time taken to complete the task, the observations made of participants' behavior during the test yields a different finding.

Their understanding of a 3D manipulation task could be seen clearly during this test. First, there is a group a people who actually used the first trial to understand how the interface worked and applied their understanding to perform better in the second trial. A few of them completed the task in less than half the time of the first attempt.

There is another group of people who got lucky during the first time and perform less well in the second trial simply because they didn't grasp the concept of 3D manipulation. Among this group, there are some who were over-eager to top their first record and ended up with a worse completion time.

The calculated mean is 65 seconds and its standard deviation is 42 seconds. Even with a drop of performance by some participants, the average completion time for the second trial does improve by 24%.

7.1.1.3 Personal Best

The participants were allowed to have as many trials as they like. A few of them stopped after the second trial. Those who continued with the next trials either wanted to keep on improving their time or wanted to get the concept right.

The first group of participants wanted to see what was the fastest time they could make. They kept on trying until they met their target. The second group of people were slightly different. They go for several trials because they did not get the concept right and kept on taking a longer time to complete the task.

	Completion Time (seconds)
1.	49
2.	71
3.	103
4.	52
5.	65
6.	67
7.	190
8.	19
9.	107
10.	127
11.	78
12.	38
13.	26
14.	51
15.	140
16.	93
17.	62
18.	157
19.	126
Mean:	85
Standard Deviation:	46

Table 7.1: The completion times of the participants during the first attempt.

	Completion Time (seconds)
1.	96
2.	31
3.	63
4.	48
5.	67
6.	22
7.	84
8.	-
9.	28
10.	37
11.	187
12.	37
13.	70
14.	76
15.	30
16.	40
17.	34
18.	135
19.	81
Mean:	65
Standard deviation:	42

Table 7.2: The completion times of the participants during the second attempt.

For this second group, I gave them the extra instructions they needed to know about manipulating the 3D object after their third trial. These instructions were about (i) how to use shadow to determine the depth of the object, (ii) how to release the object from the magnetic grip and (iii) the type of 6DoF movement needed to put the cube into the tube. Based on the information given, they were asked to repeat the task. As expected, they improved except for one participant who steadily maintained her completion time between 122 seconds to 168 seconds. Although our definition of NUI presupposes that participants can use our interface "with almost no instruction", the extra instructions given did not invalidate the results. The key piece of data we gather in our study is the completion time in the first attempt. This data point was unaffected by any extra instructions given. However, we were also interested in learning curves and personal best times; these are somewhat influenced by instructions, but these instructions are still minimal compared to, for instance, the instruction required to use Blender.

Out of all trials, I record each personal best in Table 7.3. This time, the calculated mean is 42 seconds and the standard deviation is 27 seconds. Table 7.4 summarizes the mean and the standard deviation taken from the first attempt, the second attempt and the personal best completion times.

7.1.1.4 Statistical Analysis of the Participants' Performance

Table 7.5 shows the statistical analysis that we performed based on the participants' completion time for trial 1 and 2. There were only 18 completion times because one participant did not take part in the second trial as described in Section 7.1.1.2.

Our null hypothesis is that "There was no reduction in participants' completion time in the second trial". Since each subject provides two sets of data, we performed a one-tailed, paired type T-Test, where it generated a p-value of 0.04.

The result yields $p<0.05$, which means the probability of obtaining our result by chance is less than 0.05, which is sufficient to reject our null hypothesis.

	Completion Time (seconds)
1.	25
2.	31
3.	58
4.	33
5.	45
6.	17
7.	84
8.	19
9.	28
10.	37
11.	42
12.	25
13.	26
14.	27
15.	30
16.	40
17.	34
18.	122
19.	81
Mean:	42
Standard deviation:	27

Table 7.3: The personal best completion time for each participant.

	Mean (seconds)	Standard Deviation (seconds)
First Attempt	85	46
Second Attempt	65	42
Personal Best	42	27

Table 7.4: Summary of the mean and the standard deviation for the first attempt, the second attempt and personal best.

Participants	Completion Time 1	Completion Time 2
1	49	96
2	71	31
3	103	63
4	52	48
5	65	67
6	67	22
7	190	84
9	107	28
10	127	37
11	78	187
12	38	37
13	26	70
14	51	76
15	140	30
16	93	40
17	62	34
18	157	135
19	126	81
Average:	89	64.78
Standard Deviation:	44.72	42.40

T-Test (One-tailed - paired type) - p-value:	0.04

Table 7.5: Performing the statistical analysis on the participants' completion time using T-Test.

7.1.2 The Rating

At the end of the experiment, the participants were asked to rate the interface on a scale from 1 to 5. To avoid confusion between the task and the interface itself, they were asked specifically to rate the interface based on how it was to use it without instructions. Scale 1 means it is very easy to use and scale 5 is for very hard to use.

Five participants chose scale 1, eight participants chose scale 2 and the remaining six participants voted for scale 3. From the given individual score, we calculated the average score across all ratings. This gave us scale 2 as the average, which means the interface is easy to use. Some of them reported that they chose scale 2 instead of 1 simply because they couldn't figure out straight away how the controller works until they had tried playing around with it. One of the comments made on the interface is regarding its naturalness. Although the interface is easy enough to use, the absence of haptic feedback makes it hard to determine if the cube has been properly placed inside the tube. However, in terms of the overall interface, they all agreed that the task was very enjoyable and was fun to do. In fact, the majority of them were impressed that the avatar can accurately simulate the actual Hydra controller.

Apart from the performance time and the rating score, we did not adopt any qualitative observation as these can be complicated to analyse. We focused on the objective comparison only, within the scope of our definition of a natural interface.

7.2 The Blender Interface Test

Five participants took part in the Blender user interface test. Two of them are considered expert users due to their experience. The other three are students who had used Blender for an assigment in a graphics paper.

The task is similar to the natural user interface test, which is to place each cube into its matching tube. However, as an objective comparison, we need to give supporting evidence. We do not want to compare these two interfaces in terms of the quantitative paradigm because the difference is so large, which would easily give a wrong impression. Hence, we recorded the completion time for the standard interface as well to be compared objectively with the completion time for the natural interface.

7.2.1 Expert Users

Experienced users have their own way to make things work according to their preference. In this test, both of the participants took two different approaches in performing the task.

The first participant has years of experience in using Blender, which includes production of a short animation. As a professional Blender user, he is very particular about the details of his models.

While performing the task for this experiment, he carefully placed the cubes one by one, making sure that each cube was properly aligned with its corresponding tube. His emphasis was on accuracy rather than the fastest time. For this reason, he took 545 seconds to complete this task.

The second participant had been using Blender for a couple of years and had a different way of completing the task. Since all of the tubes are tilted from the ground (pitch) at the same angle, he rotated all four cubes at once to that angle. All the cubes were then placed in front of their corresponding tubes and he manipulated each cube individually from there. He was less precise in positioning than the first participant but he managed to put all of the cubes into the tubes in 185 seconds.

He also demonstrated another way of completing the task faster. By putting up all four views - front, top, side and camera - on one screen, the task can be speeded up since he doesn't need to switch view each time he wants to align the cube with the tube. He simply dragged each cube into its tube and manipulated the cube from inside the hole without avoiding collisions. With this trick, he accomplished the same task in 51 seconds.

7.2.2 Average Users

The other three participants had the same amount of experience in using Blender. They had been introduced to the interface during a one-semester Computer Graphics paper.

As with the expert users, they were required to perform the task without any restriction on the method used. All of them did the task in the same way as the first participant in the expert category. The times taken to complete the task were 756 seconds, 522 seconds and 729 seconds.

They seem not to have much problem using the interface. The only problems they have at the beginning were to adjust themselves to the new Blender interface and to recall how to manipulate the object in Blender.

Remembering the operations is a problem for a lot of Blender users, especially beginners. Those who have been using Blender constantly will have no problem working with its interface. However, those who use Blender for a short while will not be able to fully master its interface. One of the reasons is because Blender uses different keys for different actions. The key might be used wrongly for a different operation. Another reason is the interface is less user friendly, which makes it not really easy to use.

After the experiments, they were asked to try the natural user interface. As expected, since they have been exposed to Blender, they had no difficulty interacting with the 3D space. Their completion times were 68 seconds, 84 seconds and 37 seconds.

7.3 NUI vs Blender Interface

If we look at the average completion time of the first trial of the natural interface vs the average completion time of the Blender interface, we have 85 seconds and 547 seconds respectively. The average completion time for Blender users is more than 10 standard deviations away from the average completion time for users of our interface: by any standards, Blender users would be extreme outliers in the population of our interface users, as sampled in our experiment. Even with six months training of using the Blender interface, the completion time for the first time user of our interface is still much faster. This is evidence that the interface we created is overwhelmingly better than the standard mouse-based interface for this type of manipulation.

7.4 Conclusion

There are a few conclusions that can be made based on this experiment alone. In terms of interface, the natural user interface is really easy to use because it doesn't have any command or instruction attached to it. Users just need to pick up the cube by using the controller and put it into a tube, as one would do in the real world.

The Blender interface, on the other hand, requires the users to have experience beforehand. Without knowledge of the interface, it would be almost impossible for a novice user to manipulate the object. This is something that they can't do based on instinct but they have to go through tutorials in order to understand how the interface works.

The completion times recorded for both interfaces also support our claim. If we look at the completion times of the participants during the first attempt, the slowest time recorded is still three times faster than the fastest time completed by the average Blender user.

In terms of the participants' behavior while performing the task in the natural interface, it can be said that:

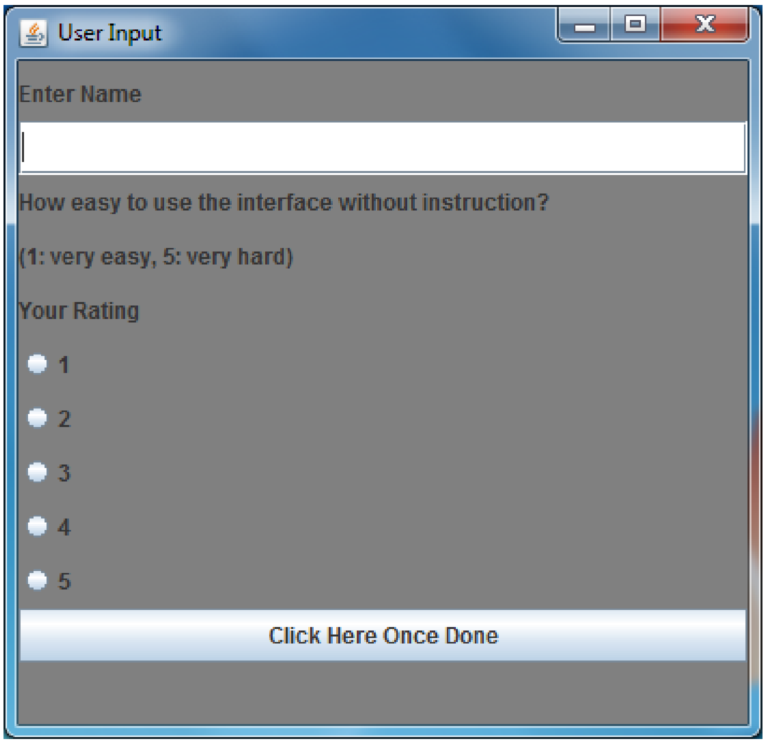
- A completion time longer than one minute during the first attempt doesn't necessarily mean that the interface is hard to use. Some of the participants who recorded more than a minute are actually using this trial to observe the interface in order to understand how it works. This is proved by the fact that they managed to record a completion time which is more than 50% faster during the second attempt.
- Many participants find it really enjoyable to beat their own completion time. However, rushing in order to gain a faster completion time often resulted in a worse completion time. This is common sense when assembling objects in the real world and the same sense is also applied to the virtual world. To gain a good result, a job should be done in a fast but calm manner. By rushing things, users are liable to make more mistakes.
- Performing a 3D task in a simulated world is not something that everybody can do straight away. However, this ability can be achieved after some training.

In Section 6.6, we listed three questions that we wanted to observe during the experiment.

- How much better are the best people than the worst people?
- Are there any people who find it very difficult?
- Is everyone able to use it successfully?

From the result of the natural user test, we could answer the above questions as follows:

- Based on the completion time of the first attempt, the best completion time is better than the worst completion time by eight times. The gap between the best and the worst time in the second trial doesn't change that much. The time difference is still about eight times greater.
- Even with the difficulty of seeing and manipulating 3D correctly, all participants are able to complete the task successfully.


User Input

Enter Name

How easy to use the interface without instruction?

(1: very easy, 5: very hard)

Your Rating

- 1
- 2
- 3
- 4
- 5

Click Here Once Done

Figure 7.1: A Java-written program to rate our natural interface.

- From the scale given to the participants at the end of the experiment, none of them rate the interface higher than 3. Furthermore, the question specifically asks the participants to rate how easy it is to use the interface without any instruction as shown in Figure 7.1. In retrospect it is quite hard to determine how to interpret this result, since there is no standard against which subjects are judging this interface. But it certainly shows these subjects do not find the interface trivially easy to use, at least in the task they were given.

Out of 19 participants, one couldn't perform the task as well as the others, even after complete instructions were given. Her performance was constant throughout all four trials, before and after the complete instructions were given. This does not prove that there is something wrong with the interface. I believe some people might require a longer training time before they can completely perform a manipulation of an object in a simulated 3D world with a complete understanding.

Based on the information gained above, we can conclude that:

- The natural interface is easier to use than the standard mouse interface.
  - Although it was found that the majority of participants did not perceive the

    simulated 3D world correctly in the natural interface, the interface is by far easier to use than the standard Blender interface.

  - The fact that all participants managed to complete the task successfully supports this claim. Participants from the average users group of the Blender interface on the other hand took quite a long time at the start of the experiment to recall the commands required to perform the manipulation task.
  - Despite the extra instructions given to some of the participants of the natural interface group, these instructions were still minimal compared to the training required by novices just to familiarize themselves with the standard modeling interface.

- The participants of the natural interface learned very quickly.
  - The participants who took time understanding the interface during the first trial were able to perform much better during the second trial based on the statistical analysis performed as shown in Table 7.5.
  - Some participants developed their understanding after performing the task more than twice.
  - Although there were participants who couldn't grasp the idea during the first few trials, after being given more instructions, they easily outperformed their previous completion time.

Looking back on the studies conducted in our evaluation, are there any **general** conclusions we can draw about how natural interfaces should be evaluated? From what we have reviewed and learned, NUI should not be defined by one interaction style only. What is natural to one interface might not be natural in another. For example, an interface designed for a virtual musical instrument will not be suitable for an assembly task interface. For a big virtual scene, using a "magic" technique to reach a far away object might increase the naturalness of the interaction, and the same "magic" technique used to select an object might look unrealistic for the interaction in a small virtual room. Our main **general** suggestion about the evolution of natural user interfaces is that it should proceed through the establishment of benchmark usability tests, which are well described and can readily be reproduced, so that the improvements to existing interfaces can be quantitatively assessed against these benchmarks. Our contribution to this general evaluation methodology is the creation of a single benchmark test in a 3D manipulation task, for our own Hydra-based interface.

In the next chapter, we wrap up the overall research findings. We also include future work that should be incorporated into this interface which we believe can increase the level of naturalness.

Chapter 8

Conclusion

8.1 Summary

In Chapter 1, we introduce the problem we want to solve, which is to create a natural 3D interface by using the existing hardware and software. This is followed by reviewing other people's work on the same area to see the interfaces that have been done. From there, we discuss about our interface in detail, from its concept to its implementation.

The research can be summarized as follows:

1. Natural interfaces are replacing GUIs.

   History repeats itself. When computer was first introduced, CLI was used as a medium of communication between a human and a machine. Later on, CLI was replaced by GUI. The current era is following the same paradigm. Natural interfaces have gained in popularity and are slowly taking over from GUI.

2. Our definition of natural interface.

   Researchers have different views on what is "natural". These differences have led to various ways of implementing the interfaces. Our focus is on creating an interface that allows a 3D object to be manipulated as if we were doing it in the real world.

   Hence, we define natural interface as:

   > *"An interface that relies almost exlcusively on visuomotor skills already possessed by users."*

3. Natural interfaces are particularly useful in 3D.

Assembly tasks are one of the examples of why natural interfaces are better than 2D traditional interfaces. It is an action that is easy to do in the real world, and yet it can be difficult to achieve with 2D interfaces.

4. Building an environment for testing 3D natural interface.

   Since an assembly task can be studied in a reasonably short experiment, we designed a task where users are required to place a cube into its matching tube by using physical hardware. This task allows users to manipulate a 3D object in 6DoF.

5. It uses only comodity (cheap) hardware.

   The designed task requires the use of physical hardware to manipulate the 3D objects. Our intention is to make use of the existing hardware to create a usable natural interface. To ensure that this project can be continued by others, we opt to use a cheap hardware, namely the Razer Hydra controller.

   The Hydra has a lot of advantages over the other popular cheap controllers such as the Wiimote and the PS Move. The use of a magnetic field to calculate the position and orientation of the controller helps in providing an accurate reading. It also allows the controller to move freely in 6DoF without noticeable latency.

6. It incorporates a physics engine.

   Implementing physics in the simulated world is important because it satisfies the physical expectations of users. A natural interface would lose its naturalism if the physics differs from the real world. Two rigid objects shouldn't penetrate each other. An object under the influence of gravity should fall normally.

   We chose the Bullet engine to handle the virtual world's physics because it is a very popular open source physics engine and has been used widely in films and commercial games. For a simple natural interaction, Bullet does the job well.

7. Calibrating the Razer Hydra.

   The Hydra controller has been calibrated to make it useful as a natural interface. We use a novel method using an average with weights calculated from a rational function of distance to correct an approximate linear transformation.

8. Conducting a usability test for the natural interface.

We conducted a simple usability test to demonstrate our idea. The participants of the natural interface test outperformed all but the most experienced users of a traditional interface and almost without training.

To measure the performance of users in different level of difficulties, we could create a different test environment by using the same interface. For example, we could design a more complicated assembly task that is used for construction such as a Lego or TinkerToy simulator.

Based on the results and the rating given by the participants, we can conclude that our natural interface is quite easy to use. Participants did seem to experience some trouble in perceiving the 3D task correctly but all of them actually managed to complete the task. There is a good evidence in the psychology literature for the difficulty of understanding seemingly simply things like cube rotations. See for example G. Hinton [49].

Overall, from what we have learnt from the experiment, we realized that for a 3D natural interface, it is very important for the users to have an understanding of the representation of the 3D scene. When designing the scene, we need to take into account the inability of most users to perceive the 3D world correctly. In designing the scene, we have to make sure that its representation does not mislead the users. For example, in our interface, the use of a kinematic object to represent the avatar often leads the user into treating the avatar as just a transparent object that can be dragged through any other static object. They did not realize that the avatar is magnetic, and attracted other dynamic objects to it, and these dynamic objects can collide with other objects in the scene.

In terms of instructions that should be given to help people to use the natural interface, we noticed that even with the training video showing the participants what they needed to do to complete the task, they still had it wrong. It is always a good idea to let them use the interface themselves, and just provide a few hints on how to control the interface. This would be a better way for them to learn to use the interface.

8.2 Future work

Our natural interface serves as a starting point towards a more ideal interface. There are many things that can be incorporated into the current interface to make it better.

As of the current state, the naturalness depends on a one to one correspondence.

Although this does give the natural impression, it can be further improved by incorporating haptic feedback into the interface.

Currently, when the cube is put into the tube, the user can tell that the task is complete through visual cues only. It would be better if we could apply some resistance when the cube collides with any part of the tube. This way, the interaction would feel more natural because the user would feel the resistance if the cube is not properly in. The use of sound to represent collision between objects - static, kinematic and dynamic - would also enhance the naturalness. When the users hear the colliding sound, they would be immediately aware that the collision has already taken place and would figure out a way to avoid the collision from happening again. Another thing that I would like to do is to make the avatar behave more like a dynamic object, which means it would collide properly with static objects in the scene, rather than just passing through.

The magnet can also be easily controlled with the use of a button. We can design a button press that will enable and disable the magnetic force. This will allow for a better control of the overall interaction.

We have not had a chance to test this application with the PS Move. Being able to compare the performance between the Hydra controller and the PS Move might lead to a different conclusion. We can experiment to see if there is any difference in performance between a wired and a wireless devices. Using the PS Move will also give an opportunity to investigate latency, whether or not it affects the usability and how to correct it.

An interface which includes an analogue of a grasping hand as well as a moving arm would be another interesting addition to our current system. However, as we mentioned earlier, hand grasping is tremendously difficult to model and is the subject of large research projects.

Our current 3D interface is implemented in a virtual space that is aligned with that in which the user's actual arm moves: the directions "forward", "left", "up" and so on in the virtual space are the same as those in the real world. Another possibility is to implement an interface in which the virtual space has some other relationship to the user's actual space. For instance, the interface could present a "mirror" arm, which moves in the way that a hand viewed in a mirror would move. This can be a very important cue in manipulating a 3D object. An object which is closer to users' hand appears to be larger, hence it gives hint as to where the manipulation should take place in the 3D space. We could also create a stereoscopic vision with motion parallax where participants can perform the 3D manipulation from different angles by moving their heads around the 3D scene.

Another good idea would be to incorporate this direct 3D manipulation into modeling software such as Blender to see if this technique can assist in the design process. We could also design the natural interface to be used for an actual assembly task such as assembling a car body parts. From here we could determine if the use of the magnetic attraction would help or hinder the 3D manipulation.

To test the participants' learning curves, the application can be designed with several levels of increasing difficulty. We could observe how the participants improve after completing each level. From the experiment, we noticed that they loved to beat their own records. Instead of competing against their own record, we could create a multiplayer platform where participants could compete against each other. This could also help to determine if the designed competition would help increasing the participants' ability to learn.

As to the evaluation component of the project: there are several interesting variants on the evaluation study which could be conducted. We could determine the improvement in performance by those who received more instructions. We could also perform a qualitative measurement, to study the reason behind the inability of most people to perform the 3D task well. Filling in questionnaires and answering open-ended questions could also help in determining the users' "feeling" while using the interface.

References

[1] Bullet Physics Library. http://bulletphysics.org/wordpress/.

[2] Sifteo Cubes. www.sifteo.com/home.

[3] Agarawala, A. and Balakrishnan, R. (2006). Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, New York, NY, USA, 1283–1292. ACM.

[4] Ahn, E., Yang, H., and Kim, G. (2012). 3D object selection for hand-held autostereoscopic display. In *Proceedings of the 10th asia pacific conference on Computer human interaction*, APCHI '12, New York, NY, USA, 59–66. ACM.

[5] Alun-Jones, B. (2010). Towards a natural user interface for visualization. Technical report, Department of Electrical and Electronic Engineering, Imperial College London.

[6] Antle, A. N., Droumeva, M., and Ha, D. (2009). Hands on what?: comparing children's mouse-based and tangible-based interaction. In *Proceedings of the 8th International Conference on Interaction Design and Children*, IDC '09, New York, NY, USA, 80–88. ACM.

[7] Baraff, D. (1991). Coping with friction for non-penetrating rigid body simulation. *SIGGRAPH Comput. Graph.*, *25*(4), 31–41.

[8] Bartoli, G., Bimbo, A. D., Faconti, M., Ferracani, A., Marini, V., Pezzatini, D., Seidenari, L., and Zilleruelo, F. (2012). Emergency medicine training with gesture driven interactive 3D simulations. In *Proceedings of the 2012 ACM workshop on User experience in e-learning and augmented technologies in education*, UXeLATE '12, New York, NY, USA, 25–30. ACM.

[9] Baumgärtner, S., Ebert, A., Deller, M., and Agne, S. (2007). 2D meets 3D: a human-centered interface for visual data exploration. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07, New York, NY, USA, 2273–2278. ACM.

[10] Baxter, B., Scheib, V., Lin, M. C., and Manocha, D. (2001). DAB: Interactive haptic painting with 3D virtual brushes. In *In Proceedings of SIGGRAPH*, New York, NY, USA, 461–468. ACM.

[11] Bier, E. A. (1987). Skitters and jacks: interactive 3D positioning tools. In *Proceedings of the 1986 workshop on Interactive 3D graphics*, I3D '86, New York, NY, USA, 183–196. ACM.

[12] Blake, J. (2012). The natural user interface revolution. In *Natural User Interfaces in .NET*. Manning Publications.

[13] Borst, C. W. (2004). Tracker Calibration using Tetrahedral Mesh and Tricubic Spline Models of War. In *Proceedings of the IEEE Virtual Reality 2004*, VR '04, Washington, DC, USA, 19–. IEEE Computer Society.

[14] Bowman, D. A., McMahan, R. P., and Ragan, E. D. (2012). Questioning naturalism in 3D user interfaces. *Commun. ACM*, *55*(9), 78–88.

[15] Bradley, D. and Roth, G. (2005). Natural interaction with virtual objects using vision-based six DOF sphere tracking. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACE '05, New York, NY, USA, 19–26. ACM.

[16] Briggs, W. (1999). Magnetic calibration by tetrahedral interpolation. In *Proceedings of NIST-ASME Industrial Virtual Reality Symposium*, 27–32.

[17] Bruder, G., Steinicke, F., and Hinrichs, K. H. (2009). Arch-Explore: A natural user interface for immersive architectural walkthroughs. In *Proceedings of the 2009 IEEE Symposium on 3D User Interfaces*, 3DUI '09, Washington, DC, USA, 75–82. IEEE Computer Society.

[18] Bryson, S. (1992). Measurement and calibration of static distortion of position data from 3D trackers. In *Proc. SPIE Conf. Stereoscopic Displays and Applications III*, 244–255.

[19] Burnett, D., Coulton, P., and Lewis, A. (2012). Providing both physical and perceived affordances using physical games pieces on touch based tablets. In *Proceedings of The 8th Australasian Conference on Interactive Entertainment: Playing the System*, Number 8 in IE '12, New York, NY, USA, 8:1–8:7. ACM.

[20] Cao, X. and Balakrishnan, R. (2003). VisionWand: interaction techniques for large displays using a passive wand tracked in 3D. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, New York, NY, USA, 173–182. ACM.

[21] Chen, M., AlRegib, G., and Juang, B.-H. (2011). An integrated framework for universal motion control. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, New York, NY, USA, 513–518. ACM.

[22] Cheng, K.-Y., Liang, R.-H., Chen, B.-Y., Laing, R.-H., and Kuo, S.-Y. (2010). iCon: utilizing everyday objects as additional, auxiliary and instant tabletop controllers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, New York, NY, USA, 1155–1164. ACM.

[23] Chittaro, L. and Sioni, R. (2012). An electromyographic study of a laser pointer-style device vs. mouse and keyboard in an object arrangement task on a large screen. *Int. J. Hum.-Comput. Stud.*, *70*(3), 234–255.

[24] Daiber, F., Falk, E., and Krger, A. (2012). Balloon selection revisited: multi-touch selection techniques for stereoscopic data. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, New York, NY, USA, 441–444. ACM.

[25] Dalsgaard, P. and Halskov, K. (2012). Tangible 3D tabletops: combining tangible tabletop interaction and 3D projection. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, New York, NY, USA, 109–118. ACM.

[26] DeCarlo, M. (2011). Xerox PARC: A Brief Nod to the Minds Behind Laser Printing, Ethernet, the GUI and More. http://www.techspot.com/guides/477-xerox-parc-tech-contributions/.

[27] Doering, T., Pfleging, B., Kray, C., and Schmidt, A. (2010). Design by physical composition for complex tangible user interfaces. In *CHI '10 Extended Abstracts on*

Human Factors in Computing Systems, CHI EA '10, New York, NY, USA, 3541–3546. ACM.

[28] Engelbart., D. Father of the Mouse. http://www.dougengelbart.org/firsts/mouse.html.

[29] Fitzmaurice, G. W., Ishii, H., and Buxton, W. A. S. (1995). Bricks: laying the foundations for graspable user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, New York, NY, USA, 442–449. ACM Press/Addison-Wesley Publishing Co.

[30] Fjeld, M., Bichsel, M., and Rauterberg, M. (1998). BUILD-IT: An Intuitive Design Tool Based on Direct Object Manipulation. In *Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction*, London, UK, UK, 297–308. Springer-Verlag.

[31] Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, New York, NY, USA, 647–656. ACM.

[32] Francese, R., Passero, I., and Tortora, G. (2012). Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, New York, NY, USA, 116–123. ACM.

[33] Froehlich, B., Hochstrate, J., Skuk, V., and Huckauf, A. (2006). The GlobeFish and the GlobeMouse: two new six degree of freedom input devices for graphics applications. In R. Grinter, T. Rodden, P. Aoki, E. Cutrell, R. Jeffriesand, and G. Olson (Eds.), *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, New York, NY, USA, 191–199. ACM.

[34] Gallo, L., Pietro, G. D., and Marra, I. (2008). 3D interaction with volumetric medical data: experiencing the Wiimote. In *Proceedings of the 1st international conference on Ambient media and systems*, Number 14 in Ambi-Sys '08, ICST, Brussels, Belgium, Belgium, 14:1–14:6. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

[35] Gascón, J., Zurdo, J. S., and Otaduy, M. A. (2010). Constraint-based simulation of adhesive contact. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Sym-*

posium on Computer Animation, SCA '10, Aire-la-Ville, Switzerland, Switzerland, 39–44. Eurographics Association.

[36] Ghazisaedy, M., Adamczyk, D., Sandin, D. J., Kenyon, R. V., and DeFanti, T. A. (1995). Ultrasonic calibration of a magnetic tracker in a virtual reality space. In *Virtual Reality Annual International Symposium, 1995*, 179–188.

[37] Ghosh, S., Zheng, J., Chen, W., Zhang, J., and Cai, Y. (2010). Real-time 3D markerless multiple hand detection and tracking for human computer interaction applications. In *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry*, VRCAI '10, New York, NY, USA, 323–330. ACM.

[38] Gorman, M. (2013). Leap Motion Controller review. http://www.engadget.com/2013/07/22/leap-motion-controller-review/.

[39] Hachet, M., Bossavit, B., Cohé, A., and de la Rivière, J.-B. (2011). Toucheo: multitouch and stereo combined in a seamless workspace. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, New York, NY, USA, 587–592. ACM.

[40] Han, S., Lee, H., Park, J., Chang, W., and Kim, C. (2010). Remote interaction for 3D manipulation. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, New York, NY, USA, 4225–4230. ACM.

[41] Hancock, M., Hilliges, O., Collins, C., Baur, D., and Carpendale, S. (2009). Exploring tangible and direct touch interfaces for manipulating 2D and 3D information on a digital table. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, New York, NY, USA, 77–84. ACM.

[42] Held, R., Gupta, A., Curless, B., and Agrawala, M. (2012). 3D puppetry: a Kinect-based interface for 3D animation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, New York, NY, USA, 423–434. ACM.

[43] Henrysson, A., Billinghurst, M., and Ollila, M. (2005). Virtual object manipulation using a mobile phone. In *Proceedings of the 2005 international conference on Augmented tele-existence*, ICAT '05, New York, NY, USA, 164–171. ACM.

[44] Henze, N., Löcken, A., Boll, S., Hesselmann, T., and Pielot, M. (2010). Free-hand gestures for music playback: deriving gestures with a user-centred process. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, MUM '10, New York, NY, USA, 16:1–16:10. ACM.

[45] Hilliges, O., Kim, D., Izadi, S., Weiss, M., and Wilson, A. D. (2012). HoloDesk: direct 3D interactions with a situated see-through display. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing System*, CHI '12, New York, NY, USA, 2421–2430. ACM.

[46] Hinckley, K., Pausch, R., Proffitt, D., and Kassell, N. F. (1998). Two-handed virtual manipulation. *ACM Trans. Comput.-Hum. Interact.*, *5*(3), 260–302.

[47] Hinckley, K. and Wigdor, D. (2012). *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (3rd ed.)., Chapter Input Technologies and Techniques, 1–49. Taylor & Francis.

[48] Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. (2010). Pen + touch = new tools. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, New York, NY, USA, 27–36. ACM.

[49] Hinton, G. (1979). Some Demonstrations of the Effects of Structural Descriptions in Mental Imagery. *Cognitive Science*, *3*, 231–250.

[50] Huang, Y. and Eisenberg, M. (2012). Easigami: virtual creation by physical folding. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, New York, NY, USA, 41–48. ACM.

[51] Huang, Y., Gross, M. D., Do, E. Y.-L., and Eisenberg, M. (2009). Easigami: a reconfigurable folded-sheet TUI. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, TEI '09, New York, NY, USA, 107–112. ACM.

[52] Hunter, S., Kalanithi, J., and Merrill, D. (2010). Make a Riddle and TeleStory: designing children's applications for the siftables platform. In *Proceedings of the 9th International Conference on Interaction Design and Children*, IDC '10, New York, NY, USA, 206–209. ACM.

[53] Ikits, M., Brederson, J. D., Hansen, C. D., and Hollerbach, J. M. (2001). An Improved Calibration Framework for Electromagnetic Tracking Devices. In *Proceedings of the Virtual Reality 2001 Conference*, VR '01, Washington, DC, USA, 63–70. IEEE Computer Society.

[54] Ishii, H. and Ullmer, B. (1997). Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, CHI '97, New York, NY, USA, 234–241. ACM.

[55] Jin, Y. K., Choi, S., Chung, A., Myung, I., Lee, J., Kim, M. C., and Woo, J. (2004). GIA: design of a gesture-based interaction photo album. *Personal Ubiquitous Comput.*, *8*(3-4), 227–233.

[56] Kadri, A., Lecuyer, A., Burkhardt, J.-M., and Richir, S. (2007). The Visual Appearance of User's Avatar Can Influence the Manipulation of Both Real Devices and Virtual Objects. In *IEEE Symposium on 3D User Interfaces.*

[57] Kaufman, D. M., Edmunds, T., and Pai, D. K. (2005). Fast frictional dynamics for rigid bodies. *ACM Trans. Graph.*, *24*(3), 946–956.

[58] Kaufman, D. M., Sueda, S., James, D. L., and Pai, D. K. (2008). Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.*, *27*(5), 164:1–164:11.

[59] Khan, A., Mordatch, I., Fitzmaurice, G., Matejka, J., and Kurtenbach, G. (2008). ViewCube: a 3D orientation indicator and controller. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D '08, New York, NY, USA, 17–25. ACM.

[60] Kim, D., Hilliges, O., Izadi, S., Butler, A. D., Chen, J., Oikonomidis, I., and Olivier, P. (2012). Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, New York, NY, USA, 167–176. ACM.

[61] Kindratenko, V. and Bennett, A. (2000). Evaluation of rotation correction techniques for electromagnetic position tracking systems. In J. D. Mulder and R. van Liere (Eds.), *Proceedings of the 6th Eurographics conference on Virtual Environments*, EG VE'00, Aire-la-Ville, Switzerland, Switzerland, 13–22. Eurographics Association.

[62] Kindratenko, V. V. (2000). A survey of electromagnetic position tracker calibration techniques. *Virtual Reality: Research, Development, and Applications*, *5*(3), 169–182.

[63] Klevjer, R. (2006). *What is the Avatar? Fiction and embodiment in Avatar-Based Single Player Computer Games*. Ph. D. thesis, University of Bergen.

[64] Kratz, S., Rohs, M., Guse, D., Müller, J., Bailly, G., and Nischt, M. (2012). PalmSpace: continuous around-device gestures vs. multitouch for 3D rotation tasks on mobile devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, New York, NY, USA, 181–188. ACM.

[65] lacolina, S. A., Soro, A., and Scateni, R. (2011). Natural exploration of 3D models. In P. Marti, A. Soro, L. Gamberini, and S. Bagnara (Eds.), *Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity*, CHItaly, New York, NY, USA, 118–121. ACM.

[66] Larson, L. (2010). CES 2010: NUI with Bill Buxton. http://channel9.msdn.com/Blogs/LarryLarsen/CES-2010-NUI-with-Bill-Buxton.

[67] Lee, D., Hwang, J.-I., Kim, G. J., and Ahn, S. C. (2011). 3D interaction using mobile device on 3D environments with large screen. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, New York, NY, USA, 575–580. ACM.

[68] Lee, J., Post, R., and Ishii, H. (2011). ZeroN: mid-air tangible interaction enabled by computer controlled magnetic levitation. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, New York, NY, USA, 327–336. ACM.

[69] Lee, J., Teerapittayanon, S., and Ishii, H. (2010). Beyond: collapsible input device for direct 3D manipulation beyond the screen. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, New York, NY, USA, 393–394. ACM.

[70] Liang, H.-N., Williams, C., Semegen, M., Stuerzlinger, W., and Irani, P. (2012). User-defined surface+motion gestures for 3D manipulation of objects at a distance through a mobile device. In *Proceedings of the 10th asia pacific conference on Computer human interaction*, APCHI '12, New York, NY, USA, 299–308. ACM.

[71] Livingston, M. A. and State, A. (1997). Magnetic tracker calibration for improved augmented reality registration. *Presence: Teleoperators and Virtual Environments*, *6*(5), 532–546.

[72] Lok, B., Naik, S., Whitton, M., and Frederick P. Brooks, J. (2003). Incorporating dynamic real objects into immersive virtual environments. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, I3D '03, 31–40. ACM.

[73] Martinet, A., Casiez, G., and Grisoni, L. (2009). 3D positioning techniques for multi-touch displays. In S. N. Spencer (Ed.), *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, New York, NY, USA, 227–228. ACM.

[74] McCutchan, J. (2011). Move.me Available Today on PlayStation Store, Free for Students and Educators. http://blog.us.playstation.com/2011/07/26/move-me-available-today-onplaystation-store-free-for-students-and-educators/.

[75] Melissen, A. (2006). Watching Window Bible. Internal report is available from the Department of Computer Science, University of Otago, on request.

[76] Mendes, D., Lopes, P., and Ferreira, A. (2011). Hands-on interactive tabletop LEGO application. In T. Romo, N. Correia, M. Inami, H. Kato, R. Prada, T. Terada, E. Dias, and T. Chambel (Eds.), *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, Number 19 in ACE '11, New York, NY, USA, 19:1–19:8. ACM.

[77] Müller-Tomfelde, C. and Schremmer, C. (2008). Touchers and mousers: commonalities and differences in co-located collaboration with multiple input devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, New York, NY, USA, 1149–1152. ACM.

[78] Naone, E. (2009). Building Blocks of a New Interface. www.technologyreview.com/article/413991/building-blocks-of-a-new-interface/.

[79] Neale, S., Chinthammit, W., Lueg, C., and Nixon, P. (2011). Natural interactions between augmented virtual objects. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference*, OzCHI '11, New York, NY, USA, 229–232. ACM.

[80] Norman, D. A. (2010). Natural user interfaces are not natural. *interactions*, *17*(3), 6–10.

[81] Oh, J.-Y. and Stuerzlinger, W. (2005). Moving objects with 2D input devices in CAD systems and Desktop Virtual Environments. In *Proceedings of Graphics Interface 2005*, GI '05, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 195–202. Canadian Human-Computer Communications Society.

[82] Pedersoli, F., Adami, N., Benini, S., and Leonardi, R. (2012). XKin -: eXtendable hand pose and gesture recognition library for Kinect. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, New York, NY, USA, 1465–1468. ACM.

[83] Perl, T. (2012). Cross-Platform Tracking of a 6DoF Motion Controller Using Computer Vision and Sensor Fusion. Master's thesis, Faculty of Informatics, Vienna University of Technology.

[84] Peters, M. and Ivanoff, J. (1999). Performance Asymmetries in Computer Mouse Control of Right-Handers, and Left-Handers with Left- and Right-Handed Mouse Experience. *Journal of Motor Behavior*, *31*(1), 86–94.

[85] Petersen, N. and Stricker, D. (2009). Continuous natural user interface: Reducing the gap between real and digital world. In *Proceedings of the 2009 8th IEEE International Symposium on Mixed and Augmented Reality*, ISMAR '09, Washington, DC, USA, 23–26. IEEE Computer Society.

[86] Raisamo, R. (1999). Multimodal human-computer interaction : a constructive and empirical study. Master's thesis, University of Tampere.

[87] Raj, M., Creem-Regehr, S. H., Rand, K. M., Stefanucci, J. K., and Thompson, W. B. (2012). Kinect based 3D object manipulation on a desktop displa. In *Proceedings of the ACM Symposium on Applied Perception*, SAP '12, New York, NY, USA, 99–102. ACM.

[88] Reisman, J. L., Davidson, P. L., and Han, J. Y. (2009). A screen-space formulation for 2D and 3D direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, New York, NY, USA, 69–78. ACM.

[89] Resl, M., Petschnigg, M., Spaelling, A., and Sommersguter, P. (2010). Comparison of Wiimote-based interaction techniques within a 3D environment. http://www.moritzresl.net/wiimote-based-interaction/.

[90] Reuter, P., Riviere, G., Couture, N., Mahut, S., and Espinasse, L. (2010). Archeo-TUI - Driving virtual reassemblies with tangible 3D interaction. *J. Comput. Cult. Herit.*, *3*(2), 4:1–4:13.

[91] Rowe, A. and Birtles, L. (2010). Glowing Pathfinder Bugs: a natural haptic 3D interface for interacting intuitively with virtual environments. In *ACM SIGGRAPH 2010 Art Gallery*, SIGGRAPH '10, New York, NY, USA, 350–358. ACM.

[92] Schlömer, T., Poppinga, B., Henze, N., and Boll, S. (2008). Gesture recognition with a Wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, New York, NY, USA, 11–14. ACM.

[93] Sharma, A., Madhvanath, S., Shekhawat, A., and Billinghurst, M. (2011). MozArt: a multimodal interface for conceptual 3D modeling. In *Proceedings of the 13th international conference on multimodal interfaces*, ICMI '11, New York, NY, USA, 307–310. ACM.

[94] Sheridan, J. G. (2011). Developing an open source exertion interface for two-handed 3D and 6DOF motion tracking and visualisation. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, BCS-HCI '11, Swinton, UK, UK, 289–298. British Computer Society.

[95] Shneiderman, B. (1987). Human-computer interaction. In R. M. Baecker and W. A. S. Buxton (Eds.), *Human-computer interaction*, Chapter Direct manipulation: A step beyond programming languages, 461–467. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[96] Sko, T. and Gardner, H. (2009). The Wiimote with multiple sensor bars: creating an affordable, virtual reality controller. In *Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, CHINZ '09, New York, NY, USA, 41–44. ACM.

[97] Smith, G., Salzman, T., and Stuerzlinger, W. (2001). 3D scene manipulation with 2D devices and constraints. In *No description on Graphics interface 2001*, GRIN'01, Toronto, Ont., Canada, Canada, 135–142. Canadian Information Processing Society.

[98] Smith, S. P., Burd, E. L., Ma, L., Alagha, I., and Hatch, A. (2011). Relative and absolute mappings for rotating remote 3D objects on multi-touch tabletops. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, BCS-HCI '11, Swinton, UK, UK, 299–308. British Computer Society.

[99] Song, P., Goh, W. B., Hutama, W., Fu, C.-W., and Liu, X. (2012). A handle bar metaphor for virtual object manipulation with mid-air interaction. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, New York, NY, USA, 1297–1306. ACM.

[100] Sreedharan, S., Zurita, E. S., and Plimmer, B. (2007). 3D input for 3D worlds. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, New York, NY, USA, 227–230. ACM.

[101] Strothoff, S., Valkov, D., and Hinrichs, K. (2011). Triangle cursor: interactions with objects above the tabletop. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, New York, NY, USA, 111–119. ACM.

[102] Sugano, Y., Harada, K., and Sato, Y. (2012). Touch-consistent perspective for direct interaction under motion parallax. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, ITS '12, New York, NY, USA, 339–342. ACM.

[103] Sutherland, I. E. (1963). *Sketchpad, A Man-Machine Graphical Communication System.* Ph. D. thesis, Massachusetts Institute of Technology, United States.

[104] Tanaka, K., Parker, J., Baradoy, G., Sheehan, D., Holash, J. R., and Katz, L. (2012). A Comparison of Exergaming Interfaces for Use in Rehabilitation Programs and Research. *Loading...*, *6*(9), 69–81.

[105] Venolia, D. (1993). Facile 3D direct manipulation. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, New York, NY, USA, 31–36. ACM.

[106] Wang, R. Y., Paris, S., and Popovic, J. (2011). 6D hands: markerless hand-tracking for computer aided design. In J. S. Pierce, M. Agrawala, and S. R. Klemmer (Eds.), *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, New York, NY, USA, 549–558. ACM.

[107] Watanabe, R., Itoh, Y., Asai, M., Kitamura, Y., Kishino, F., and Kikuchi, H. (2004). The soul of ActiveCube: implementing a flexible, multimodal, three-dimensional spatial tangible interface. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACE '04, New York, NY, USA, 173–180. ACM.

[108] Wingrave, C. A., Williamson, B., Varcholik, P. D., Rose, J., Miller, A., Charbonneau, E., Bott, J., and Jr., J. J. L. (2010). The Wiimote and Beyond: Spatially Convenient Devices for 3D User Interfaces. *IEEE Computer Graphics and Applications*, *30*(2), 71–85.

[109] Withana, A., Kondo, M., Makino, Y., Kakehi, G., Sugimoto, M., and Inami, M. (2010). ImpAct: Immersive haptic stylus to enable direct touch and manipulation for surface computing. *Comput. Entertain.*, *8*(2), 9:1–9:16.

[110] Witten, I. H. and Greenberg, S. (1993). User Interfaces. In A. Ralston and E. Reilly (Eds.), *Encyclopaedia of Computer Science,*, 1411–1414. New York: Van Nostrand Reinhold.

[111] Wong, E. L., Yuen, W. Y. F., and Choy, C. S. T. (2008). Designing Wii controller: a powerful musical instrument in an interactive music performance system. In G. Kotsis, D. Taniar, E. Pardede, and I. Khalil (Eds.), *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '08, New York, NY, USA, 82–87. ACM.

[112] Wu, A., Reilly, D., Tang, A., and Mazalek, A. (2011). Tangible navigation and object manipulation in virtual environments. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, New York, NY, USA, 37–44. ACM.

[113] Wyvill, G. and Philips, R. (2011). The Octagon. In *Proceedings of the 11th SIGGRAPH Asia 2011 on Emerging Technologies*, Number 8. ACM New York.

[114] Yang, R. S., Strozzi, A. G., Lau, A., Lutteroth, C., Chan, Y. H., and Delmas, P. (2012). Bimanual natural user interaction for 3D modelling application using stereo computer vision. In *Proceedings of the 13th International Conference of the NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, CHINZ '12, New York, NY, USA, 44–51. ACM.

[115] Yin, Y. and Davis, R. (2010). Toward natural interaction in the real world: real-time gesture recognition. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, Number 15 in ICMI-MLMI '10, New York, NY, USA, 15:1–15:8. ACM.

[116] Yoo, B., Han, J.-J., Choi, C., seob Ryu, H., Park, D. S., and Kim, C. Y. (2011). 3D remote interface for smart displays. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, New York, NY, USA, 551–560. ACM.

[117] Zachmann, G. (1997). Distortion Correction of Magnetic Fields for Position Tracking. In *Proceedings of the 1997 Conference on Computer Graphics International*, CGI '97, Washington, DC, USA, 213–220. IEEE Computer Society.

[118] Zachmann, G., Ehnes, J., Giersig, A., Zieringer, A., and Do, V. (2000). Virtual Reality in Assembly Simulation - Collision Detection, Simulation Algorithms, and Interaction Techniques. Darmstadt: Technischen Universitat Darmstadt.

[119] Zhang, L., Shi, Y., and Fan, M. (2008). UCam: direct manipulation using handheld camera for 3D gesture interaction. In *Proceedings of the 16th ACM international conference on Multimedia*, MM '08, New York, NY, USA, 801–804. ACM.

[120] Zhao, N. (2006). A GUI for Biomechanical Modelling and Simulation. Master's thesis, University of Otago.

[121] Zigelbaum, J., Kumpf, A., Vazquez, A., and Ishii, H. (2008). Slurp: tangibility spatiality and an eyedropper. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, New York, NY, USA, 2565–2574. ACM.

[122] Zizka, J., Olwal, A., and Raskar, R. (2011). SpeckleSense: fast, precise, low-cost and compact motion sensing using laser speckle. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, New York, NY, USA, 489–498. ACM.