NTNU – Trondheim
Norwegian University of
Science and Technology

Petrel Software Usability Study: Using the Microsoft Kinect v2 for Workflow Execution

Christopher Benjamin Westlye

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Amund Skavhaug, ITK
Co-supervisor: Ahmed Adnan Aqrawi, Schlumberger Information Solutions

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Assignment

This master's project was requested by the company Schlumberger and the Institute for Engineering Cybernetics at NTNU. Set up as a feasibility study to explore the Kinect as an interaction device for seismic interpretation, the project set out to accomplish the following:

- Study relevant background theory on interaction design and usability testing
- Study existing interaction solutions
- Analyze the results from the previous project (Christopher Benjamin Westlye (2014))
- Identify new interaction possibilities for the Kinect based on the studied material
- Plan and implement Kinect support for the Petrel software acting as the commercial testing software
- Conduct a usability study with the aforementioned application
- Analyze the Kinect's potential as a feasible interaction device for executing realistic Petrel workflows

Trondheim, 2015-06-17

Christopher Benjamin Westlye

Abstract

The computer mouse has been one of the most commonly used interaction devices for a long time. This thesis set out to explore new interaction possibilities in regards to effectiveness, efficiency and satisfaction. The Microsoft Kinect v2 sensor and the traditional computer mouse were selected for comparison. Kinect support was implemented for Petrel software and a usability study was conducted to evaluate Kinect's feasibility. The metrics used for evaluating usability were effectiveness, efficiency and satisfaction as reported by test participants through use of the think-aloud method and questionnaires. The test consisted of three parts, one about getting familiar with the software, one about data optimization and one about data visualization.

Ten students in the age range 21-29 participated in the study. 70% had used a Kinect before while only 10% had used Petrel software before. The test was performed first with a mouse and afterwards with the Kinect. Participants were required to provide feedback both during and after testing, both vocally and in writing. A SUS questionnaire was used, with mouse scores being 2.2% higher on average than Kinect scores. Similarly close results were seen in execution times, with only one of the three test parts differing significantly between devices. Task two took one minute and 24 seconds longer on average with Kinect. It can be concluded that the precision of an implemented gesture is essential for the Kinect's performance.

The qualitative analysis suggested that users preferred the mouse in regards to effectiveness and efficiency, but preferred the Kinect in regards to user satisfaction. It is likely that this is due to the Kinect's novelty effect. Both numerical results and the user evaluations were closer in value than anticipated. The findings implicate that supporting the right gestures and providing the required precision could make the Kinect a competitive interaction device in some areas. The most common suggestions made by study participants were demonstrations and presentations. It can be concluded that the Kinect is a good alternative where the fun and novelty aspects are important.

Sammendrag

Datamus har vært en av de mest brukte interaksjonsenhetene i lang tid. Denne oppgaven satte ut for å utforske nye interaksjonsmuligheter i forhold til effektivitet, ytelse og brukertilfredshet. Microsoft Kinect v2 sensoren og den tradisjonelle datamusen ble valgt for sammenligning. Kinect-støtte ble implementert for Petrel programvare og en studie i brukervennlighet ble utført for å evaluere Kinect. Hovedmålene som ble brukt for å vurdere brukervennlighet var effektivitet, ytelse og brukertilfredshet som rapportert av testdeltakerne gjennom bruk av tenkehøyt-metoden og spørreskjemaer. Testen besto av tre deler, en som handlet om å bli kjent med programvaren, en om optimalisering av data og en om visualisering av data.

Ti studenter i alderen 21-29 deltok i studien. 70% hadde brukt en Kinect før mens bare 10% hadde brukt Petrel programvare før. Testen ble først utført med en mus og etterpå med Kinect. Deltakerne ble bedt om å gi tilbakemelding både under og etter testing, både vokalt og skriftlig. Et SUS spørreskjema ble brukt, der datamusens gjennomsnittlige score var 2,2% høyere enn Kinect sin. Fullføringstidene var også nær hverandre i verdi for del en og tre av testen. Bare tidsbruken for del to avvek betydelig mellom enhetene. Del to tok i gjennomsnitt ett minutt og 24 sekunder lengre med Kinect. Det kan konkluderes med at presisjonen av en implementert gest er viktig for Kinect ytelse.

Den kvalitative analysen antydet at brukerne foretrakk musen i forhold til effektivitet og ytelse, men foretrakk Kinect i forhold til brukertilfredshet. Det er sannsynlig at dette er på grunn av brukernes begrensede erfaringer med Kinect, noe som gjøre den til en spennende enhet. Både numeriske resultater og brukerevalueringer var nærmere i verdi enn forventet. Funnene implisererer at det å støtte de riktige bevegelser og gi den nødvendige presisjonen kan gjøre Kinect til en konkurransedyktig enhet på enkelte områder. De vanligste forslagene fra deltagerne var demonstrasjoner og presentasjoner. Det kan dermed konkluderes med at Kinect er et godt alternativ der de viktigste aspektene er morsomme og interessante interaksjoner.

Acknowledgment

First of all I would like to thank both of my supervisors for all their help. Amund Skavhaug has provided invaluable feedback on both the execution and final results of this project. Secondly, Ahmed Adnan Aqrawi at Schlumberger has given amazing support and expertise throughout the entire project. Both advisors have been an inspiration with their enthusiasm and knowledge.

I would also like to thank the company Schlumberger for giving me the opportunity to execute this project by providing much needed software, hardware, guidance and expertise.

Last but not least, I would like to thank the people I share an office with. Having someone to talk to when you are stuck on a problem is immensely useful both for solving the task and on an emotional level. Not to mention the joy of having someone nearby to share coffee breaks with.

C.B.W.

Contents

Chapter 1

Introduction

The beginning of this chapter is an improved version of the introduction in my previous thesis (Christopher Benjamin Westlye (2014)).

The Field of Computer Science

The field of Computer Science has been in constant development ever since general purpose computers were introduced. Since the emergence of personal computing, society has changed in regards to how people do their work, communicate with each other and generally spend their time. As both software and hardware improve however, a number of challenges appear. As software products are updated and expanded upon to take advantage of increasing processing power and cover user needs, more and more features are added. Unless cared for, this causes what Hsi and Potts (2000) refers to as *creeping featurism*, causing visual clutter and difficulties when navigating the software. It quickly becomes clear that simplicity of design and usability is paramount for both the sale of a product and the efficiency at which the user is able to take advantage of the program. The same should hold for hardware. But despite advances in technology, the computer mouse is still the most common tool for interacting with computers.

Graphical Interface

The graphically presented information on the screen has also undergone continuous development. Today's colorful WIMP (Windows, Icons, Menus and Pointers) interfaces are a far cry from the old black and white terminals. As mentioned in the previous section, the availability of col-

orful graphical displays and powerful hardware should allow for more intuitive ways to interact with computers.

Interaction Methods

The combination of keyboard and mouse has been a staple for interacting with computers for a long time. Despite attempts at changing this pattern, touch screens are probably the only prominent alternative of today. While the computer mouse is undoubtedly precise, it has its issues. If a toddler is given a computer mouse and a touch pad, the toddler will have an easier time learning to use the touch pad. The pad simulates natural environment interaction by responding to touch at the touched location. So even though the screen interface might make little sense to the toddler, touching a particular area will activate that area. Comparatively, the computer mouse is less intuitive. Though precise, movement of the mouse corresponds to the cursor moving across the computer screen. This discontinuity between area of touch and area of action dictates that interaction with mouse is to a larger extent a learned behavior. Could an interaction method more closely related to how humans interact with their environment boost not only interaction learning, but efficiency as well?

Increasing Computational Power

As shown in Figure 1.1 and well known through Moore's law (Moore (1965)), computer power has been increasing rapidly. Whether this holds up is subject to debate, but the current evolution allows today's users higher fidelity and more accurate ways of human-computer interaction. Despite Wirth's law (Wirth (1995)) about hardware power being negated by increases in software requirements, the graph's portrayal of development should enable use of interfaces as imagined in movies like Minority Report and the Iron Man series. This project is about exploring the potential of a gesture based interaction device like the Kinect. Figure 1.2 shows an example of a program where the interface is susceptible to clutter. Could a more natural interaction method be effective in such a setting?

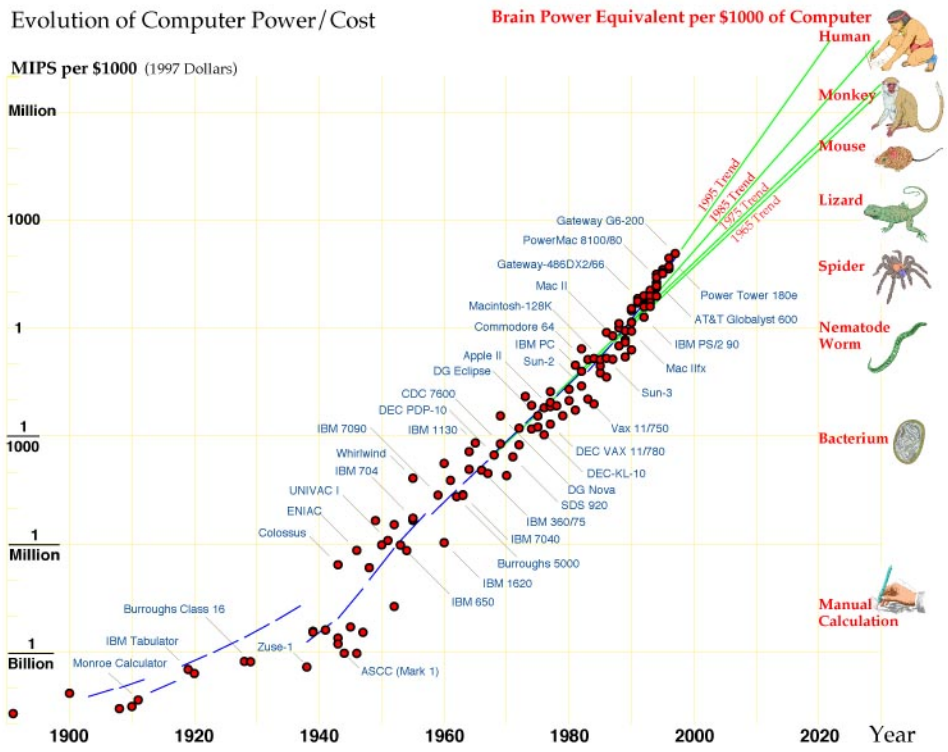


Figure 1.1: Computer power vs. cost over time

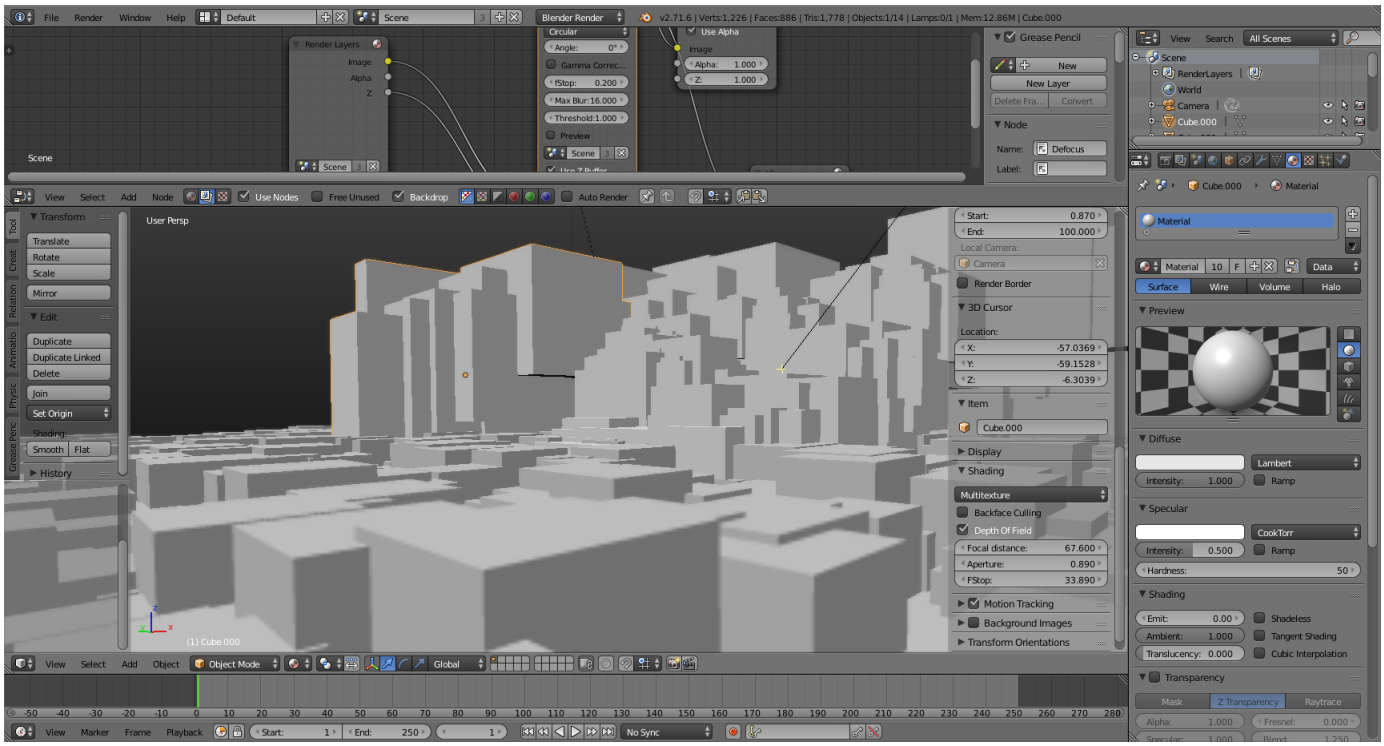


Figure 1.2: Arbitrary 3D Modeling Application

Problem Formulation

The main objective in this project is to compare the well established computer mouse with a lesser known alternative. The Microsoft Kinect for Windows v2 was chosen as the comparison device. A Kinect implementation will be developed and a study conducted. The goal is to analyze the possibilities and weaknesses of each device. The gesture based approach of the Microsoft Kinect opens up many different ways to develop our test interactions. The implementation will therefore be based on recommended gestures and guidelines in existing publications on the subject, notably Microsoft (2014). Gestures are also made with results from previous project work in mind (Christopher Benjamin Westlye (2014)). Both qualitative and quantitative results will be measured and compared.

1.1 Objectives

The main objectives of this project are shown in the list below.

1. Develop a Kinect implementation that enables effective and precise interaction
2. Perform a qualitative and quantitative user study with a focus on effectiveness, efficiency and user satisfaction
3. Analyze the results

1.2 Limitations

Available Time

Before even beginning with the implementation, access is needed to relevant equipment. Experience has shown that acquiring a computer that is compatible with both Petrel and the Kinect might be problematic. Test implementation will also use up a good amount of project time. With this in mind, the test is designed to look at broad interaction genres and not small differences, for example between whether or not the hand gestures are done in front of or beside the user's body. With more time the amount of test participants could also have been increased, as well as

the level of detail that is tested. The study is limited to see if the Kinect excels at any particular level of required interaction precision.

1.3 Approach

The following is a short description about how each of the object entries given in section 1.1 is approached.

Develop a Kinect implementation that enables effective and precise interaction

The first thing to do is conduct a thorough study of both existing usability studies and literature regarding gesture based interaction design. The previous project also needs to be thoroughly reviewed. After having a basic understanding, the supported gestures and the related Kinect implementation will be designed to conform with the studied literature.

Perform a qualitative and quantitative user study with a focus on basic interaction

Before testing the interaction designs, the test study needs to be prepared. Questionnaires and required tools must be understood so that the testing can be performed in a professional manner. The participants should not be affected by the test moderator not being prepared for unforeseen events. A suitable location is also important.

Analyze the results

While the report should be a continuous work throughout the project, the final part of the work process is filling out the results and analyzing them in the context of the usability comparison that is the goal of this project.

1.4 Disposition

The rest of the report is organized as follows:

Chapter 2 gives an introduction to relevant background material for this project, allowing the reader to understand the presented work. The Microsoft Kinect is presented in detail in addition to design principles for natural user interfaces. Other relevant software is also mentioned.

Chapter 3 provides an explanation for the methodology followed during the testing. How the environment was set up, what was tested, how it was tested and who was tested are subjects for this chapter. Relevant literature is also described.

Chapter 4 describes the Kinect implementation. Pictures and design decisions are presented.

Chapter 5 discusses the study results. Qualitative and quantitative results are presented and analyzed.

Chapter 6 attempts to make conclusions based on the result analysis. Recommendations for future work is also provided.

Chapter 2

Background

This chapter introduces existing theory used in this project. Sections 2.2, 2.3and 2.5 are taken from the background chapter in my previous thesis (Christopher Benjamin Westlye (2014)) as there was no need to rewrite the same theory twice. The intention with this chapter is to provide a theoretical background for understanding the testing and analysis performed. The reader should therefore feel free to skip familiar sections. Section 2.1 starts of by introductin the concept of usability. In section 2.2 the Microsoft Kinect is presented both technically and historically to shed some light on as to why it is suitable for this project. Section 2.3 presents the Kinect's SDK and why it was chosen over other available SDK's. Section 2.4 presents the Petrel software which was used for the testing in this project. Finally, section 2.5 talks about the definition of a Natural User Interface (NUI) and how it differs from other interface types.

2.1 Usability

2.1.1 Definition

To be able to properly conduct a usability study, it is important to understand what the term usability entails. While there is no singular definition, there seem to be some concepts that are widely agreed upon. It is the details of each definition that seem to differ.

Standard Definitions

The following are broad definitions given by some widely cited sources.

- *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"* (International Organization For Standardization (1998))
- *"Usability is a **quality attribute** that assesses how easy user interfaces are to use. The word "usability" also refers to methods for improving ease-of-use during the design process"* (Nielsen (2012))
- *"Usability is not a quality that exists in any real or absolute sense. Perhaps it can be best summed up as being a general quality of the **appropriateness to a purpose** of any particular artefact"* (Brooke (1996))

Usability Components

While the definitions above are similar, the proposed ways for measuring usability differs from source to source. The ISO 9241-11 standard (International Organization For Standardization (1998)) defines the main usability measures as:

- ***Effectiveness:*** *"accuracy and completeness with which users achieve specified goals"*
- ***Efficiency:*** *"resources expended in relation to the accuracy and completeness with which users achieve goals"*
- ***Satisfaction:*** *"freedom from discomfort, and positive attitudes towards the use of the product"*

Jakob Nielsen's definition share some similarities (Nielsen (2012)):

- ***Learnability:*** *How easy is it for users to accomplish basic tasks the first time they encounter the design?*
- ***Efficiency:*** *Once users have learned the design, how quickly can they perform tasks?*

- ***Memorability:*** *When users return to the design after a period of not using it, how easily can they reestablish proficiency?*
- ***Errors:*** *How many errors do users make, how severe are these errors, and how easily can they recover from the errors?*
- ***Satisfaction:*** *How pleasant is it to use the design?*

There is a high amount of overlap between the usability components suggested by either author. This thesis and the accompanying test focuses mainly on the ISO standard components. Effectiveness, efficiency and satisfaction.

2.1.2 Why Are Usability Studies Important?

Usability studies are important for many reasons. If a commercial product has low usability, people will stop using it and go over to competitors. In a work environment with for example Petrel, usability is important with regards to productivity. The more effective a program is at completing the tasks it is meant for, the more productive it's use will be. Usability studies are a way of identifying an interface's usability and by extension it's performance in the hands of a person.

2.1.3 Usability Characteristics of the Problem

The relevant interfaces in this thesis are physical human-machine interfaces. Both the mouse and the Kinect require a person's body to move for interaction to happen, making efficiency a key factor. If an interface requires a lot of movement for basic tasks, it will end up tiring a person and reduce productivity by extension. The mouse is of course well-established and efficient at what it does. So to mirror this in the Kinect, the focus is on gestures being efficient. The interface might never end up with the same level of efficiency as the mouse, so another important element of this thesis is identifying areas where the Kinect's strength might make it a feasible interface.

Figure 2.1: Kinect for Windows v2. Picture taken from Microsoft's website

2.2 Microsoft Kinect

2.2.1 General Information

The Microsoft Kinect is a peripheral device with several sensors, shown in figure 2.1. Its inputs are an RGB camera, a depth sensor and a multi-array microphone. Using these sensors, the Kinect captures video and audio data about what is going on in front of it. The captured data is transferred to a computer, either a traditional one or an Xbox (Microsoft). The data is processed in the receiving device, resulting in visual and auditive information with high levels of abstraction. As an example, the depth sensor returns depth levels for all surfaces in front of the sensor. This is used to identify potential users in front of the Kinect, and their movements.

Popularity

It might seem like the Kinect is losing popularity. Microsoft was recently pressured by public demand to provide the new Xbox One without the Kinect, as many customers did not want the

device. Despite this, Kinect has definitely established itself as a popular device in developer communities and academia. The official SDK provided by Microsoft and detailed in section 2.3 is a proof of this, and Microsoft is definitely using resources to keep the interface updated, further detailed in 2.3.3. As a measure of popularity, the Kinect for Xbox 360, which was the first released version, achieved a Guinness World Record for selling 133 330 units on average per day for the first 60 days (approximately 800 000 units in 60 days) (Records). This popularity with both regular users and developers is one of the reasons the Kinect was chosen for this project.

The Kinect as a Natural User Interface

As mentioned in the previous section, the Kinect provides a natural user interface by capturing gesture and voice data from up to multiple people in it's field of view. This allows computer interaction without the need to touch an external device, the most common and relevant example of this being a computer mouse. Depending on the effectiveness of the implementation, the gestures can be made as similar as possible to the way a user would interact with an object in the real world. This lack of a physical device is what makes the Kinect a natural user interface, as described in section 2.5. This makes it a possibility to develop a naturally intuitive interface that potentially requires little training before the user is capable of efficient interaction.

2.2.2 History

The Microsoft Kinect, during its development referred to as Project Natal, was originally developed for use with Microsoft's home entertainment system Xbox 360. The first version was released in November 2010, and was meant to make video games appeal to a broader population than the typical gamer. A version was released for Windows in February 2012. The device had been used by developers before a version was released specifically for the Windows OS, so the official SDK release was more like an acknowledgment and desire from Microsoft to support third party development.

Versions

The currently existing versions of the Kinect are shown below. Earliest worldwide release date is also provided.

1. Kinect for Xbox 360 (November 4, 2010)
2. Kinect for Windows v1 (February 1, 2012)
3. Kinect for Xbox One (November 22, 2013)
4. Kinect for Windows v2 (July 15, 2014)

The Xbox 360 and the Windows v1 version are technically similar. The same can be said about the Xbox One and Windows v2 versions. The main difference between the similar versions is the USB adapter. The Xbox versions have a single wire for both power and data transmission, while the Windows versions have separate wires. This means that the versions can be used interchangeably, given that one has access to the extended adapter for the Windows versions.

Version for This Project

As mentioned in the previous section, the newest version is the Kinect for Windows v2. This is the one that was used in this project. If nothing else is specified, this is the version that is talked about in the thesis.

2.2.3 Technical Specifications

The operating resolution and frequency for the sensors relevant to the project are shown in table 2.1. The main sensor is the depth sensor. Consisting of three infrared emitters and a camera able to see the returning infrared light, the depth sensor uses a time-of-flight approach to capture depth data. Measuring the return time values of each point in the field of view, the relative positions of present objects are made clear. This can then further be processed to identify people and their movements. Figure 2.2 shows the range and field of view for the Kinect. This is an improvement over the older versions, and allows interaction in a normal office environment.

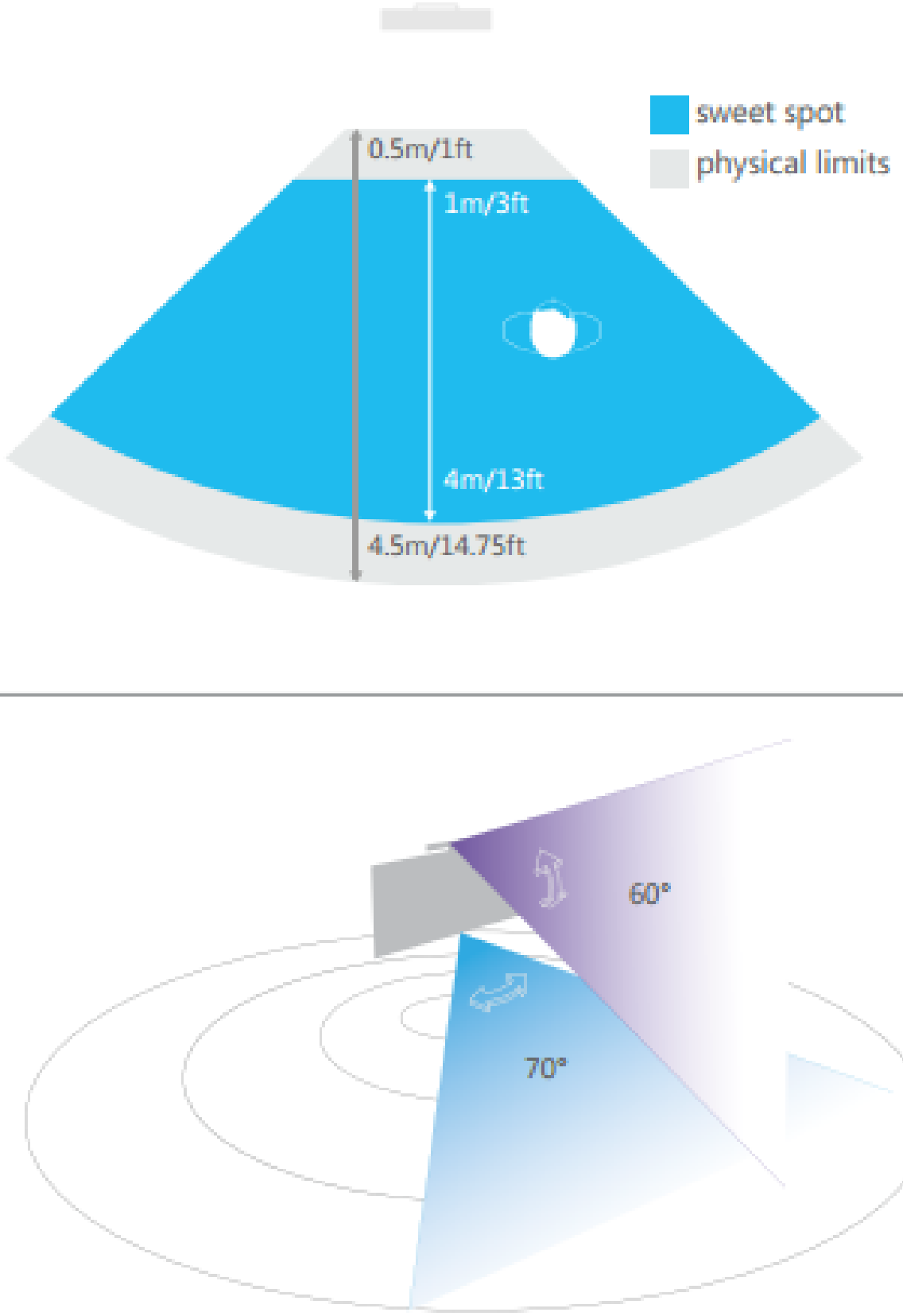


Figure 2.2: Optimal range and field of view for the Kinect v2. Picture taken from Microsoft (2014)

Sensor Type	Resolution	Frequency
RGB Camera	1920 x 1080	30 Hz (15 Hz low light)
Depth Sensor	512 x 424	30 Hz

Table 2.1: Data taken from Microsoft's website

2.3 Kinect for Windows SDK 2.0

The official software development kit from Microsoft was used for this project. The newest version is presented here.

2.3.1 APIs

The SDK includes three supported application programming interface sets (API). The APIs can all be used to create applications with Kinect support.

Native APIs

The native APIs are included for applications that require the performance capabilities of native code. C++ is supported, but as our project is not particularly power intensive, a higher level API was chosen.

.NET Framework APIs

This was the API set used for the developed test application. Windows platform foundation (WPF) development is supported, making graphical interfaces programmable with the use of XAML and C# code. Useful for general application development.

Windows Runtime APIs

The final set of APIs support Windows Store App development for Windows 8.

2.3.2 Kinect Data Sources

These are the different data types that can be extracted from the Kinect. The data is accessed by subscribing to events that fire when new data is available. Polling the Kinect is also possible.

Color

The color source provides image data from the Kinect's RGB camera.

Depth

The depth source provides depth data about each point seen by the Kinect's infrared receiver.

Body

The body source uses the depth data to find people in the Kinect's field of view. Once a person is found, the 3D position of joints are grouped up and provided through the source. This is the main source used for this project, as it provides a high level of abstraction.

Body Index

Provides data similar to the body source. The focus in this source is not body joints, but rather to identify which points are part of people's bodies and which are background.

Infrared

This source provides image data from the Kinect's infrared camera sensor.

2.3.3 Applications

The main applications included with the SDK are presented here.

Kinect SDK Browser

The SDK Browser is a simple gateway application with links to included programs, computer requirements checking and several code examples using the supported APIs.

Kinect Studio

Kinect Studio let's the user specify which sources to view from the Kinect in a live stream. There is also the possibility of capturing data to be used for testing or in the next application to be mentioned here.

Visual Gesture Builder

The Visual Gesture Builder is a program that, given input data recorded in Kinect Studio, can identify desired gestures. Based on these gestures, machine learning is used to provide a way to program the gesture without using a heuristic approach and writing the identification code manually. This is a huge benefit as compared to having to model the gestures using a heuristic approach.

2.4 Petrel

2.4.1 About

Petrel was originally published in 1996 as a tool for seismic simulations. The development of the program was related to the more and more specialized geoscientist work environment. Having been rigorously worked on since then, Petrel is today considered to be an integrated work flow tool that includes most reservoir modeling procedures. This allows specialized scientists to work together in an efficient manner. Petrel is also considered to be the biggest exploration and production software platform in the oil and gas industry.

2.4.2 User Interface

Figure 2.3 shows the Petrel interface. It was recently updated to feature a ribbon based user interface instead of the classic interface. The interface is based around two main windows. The display window and the explorer panes. The explorer panes can be seen on the left side of the screen. One of the explorer panes is the input pane, which is relevant for the usability study in this project. Here, all imported and created data files are listed. This includes the 3D seismic model that was used for testing. The display window is where objects are displayed. Many different types of windows can be opened. Only 3D windows are used in this project, allowing the user to observe seismic data in a 3D environment. In addition to the two main windows, Petrel features numerous menus and options. As shown in figure 2.3, the menu bar is at the top of the screen. It is categorized into different tabs. Also worth mentioning is the window toolbar, shown just above the 3D seismic model in the figure. It is featured quite heavily in the test. Table

Version	Petrel 2015.1 64bit
Build Date	Jan 16 2015
Build No.	P581707-1847

Table 2.2: Petrel in this Project

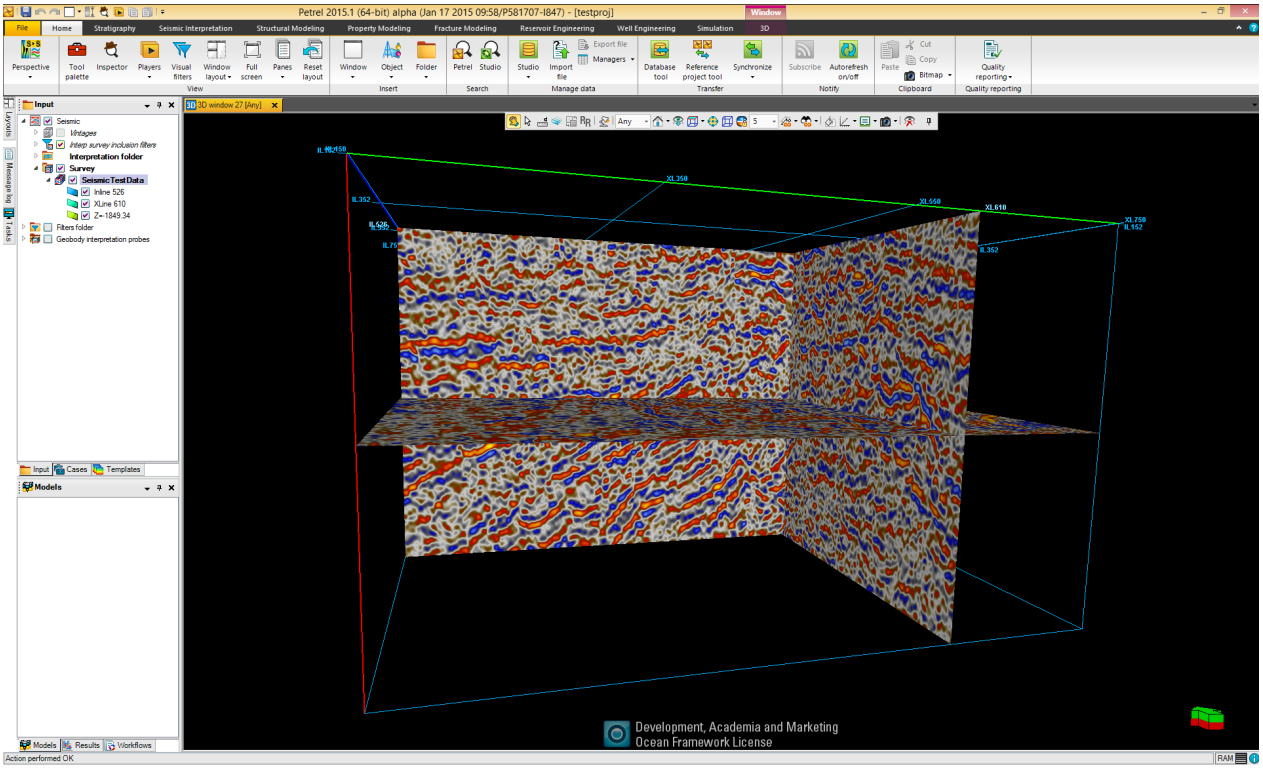


Figure 2.3: Petrel Ribbon Interface

2.2 shows details about the Petrel version used in this project.

2.5 Natural User Interfaces

NUI is a common term for interfaces that are more or less invisible to the user. The user is able to interact with a machine without the need to learn and adapt an artificial set of skills. An NUI is usually linked with a low entry barrier and a similarity to real world interactions. The Kinect exemplifies this by allowing hand gestures as a valid computer input. Using hand gestures to interact with others is a common part of human society, so by using this fact when designing computer interaction, there could be potential for a more efficient and natural user input, as pointed out by Pavlovic et al. (1997).

Chapter 3

Methodology

To explore Kinect's feasibility as an interaction device for Petrel, a usability test with ten participants was conducted. For the results to hold relevance, proper methodology is required. In this chapter, the different aspects and considerations of the testing method are described. In section 3.1, the testing environment where the study was performed is described. In section 3.2, the tasks given to the users are presented. Section 3.3 explains the methods and questionnaires used to evaluate the test participants. Finally, section 3.4 explains the reasoning behind participant selection.

3.1 The Environment

3.1.1 Environment Selection

According to Rubin and Chisnell (2008), it is not necessarily ideal for a usability test to be performed in a laboratory setting. Some organizations invest in expensive and well-equipped laboratories designed for usability testing. An example of such an environment is shown in figure 3.1. If this approach is accompanied by proper design methodology it might be useful, but it often fails to take into account that usability is more about design philosophy and less about fancy test setups. The selection of environment should be linked with the design of the study, test application and target group. This is so that the results hold relevance for the intended application of the product. With this in mind, the project was performed in a closed office room.

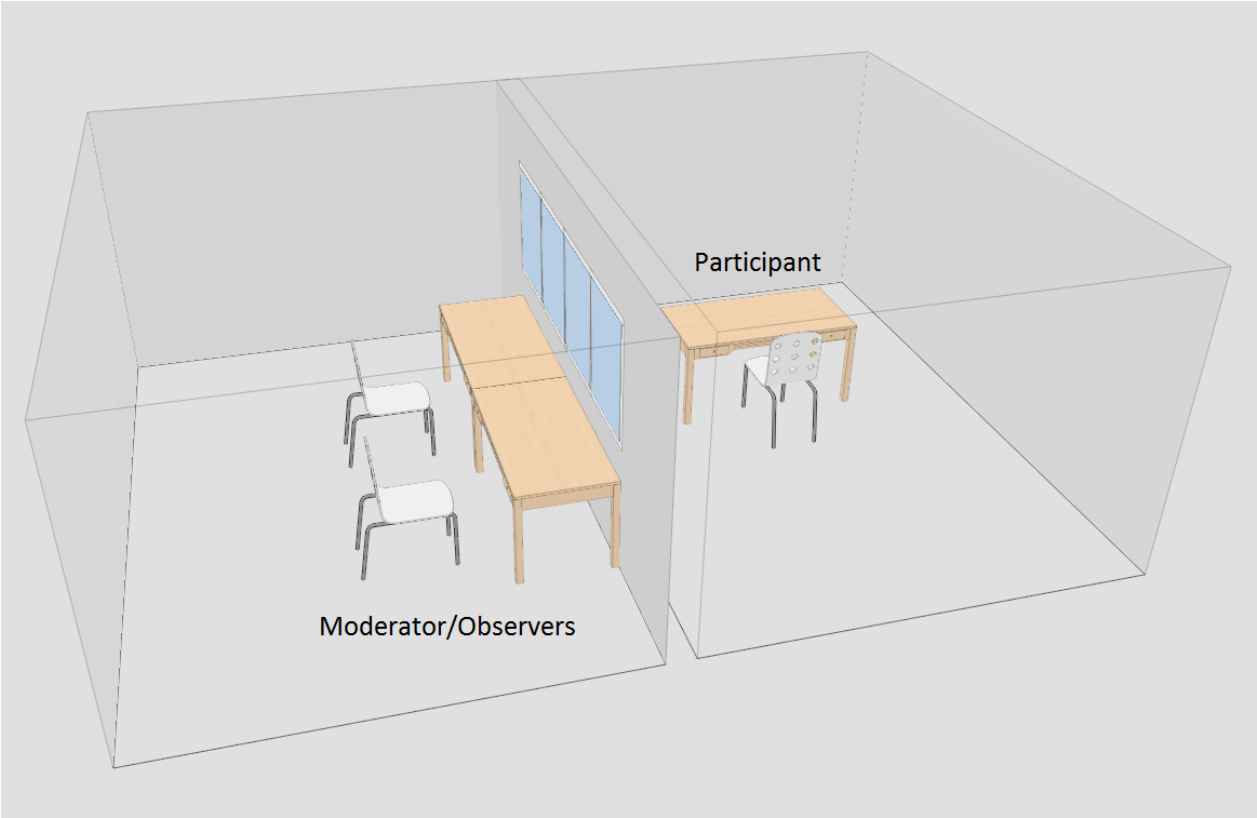


Figure 3.1: Laboratory Example Setup. Made with IKEA Home Planner

Figure 3.2 shows the general layout of the environment. This project is about using the Kinect in an office environment, and so it seemed natural to perform the testing in such a setting.

Rubin and Chisnell (2008) also details several test setups, with *Simple Single-Room Setup* being the name given for the type of environment used here, with the addition of the other people in the room. This is the most basic type of testing environment. All that is required is a quiet, secluded room with the necessary testing equipment. Picture 3.2 depicts the testing area. The test subject was put in front of the computer interface with the test moderator close by, as shown.

3.1.2 Relaxed Environment

External elements that were deemed distracting, such as hardware related to other projects, were removed from the testing area. Ordinary everyday items were kept around to prevent the environment from becoming sterile. Only attention-grabbing elements were removed to pre-

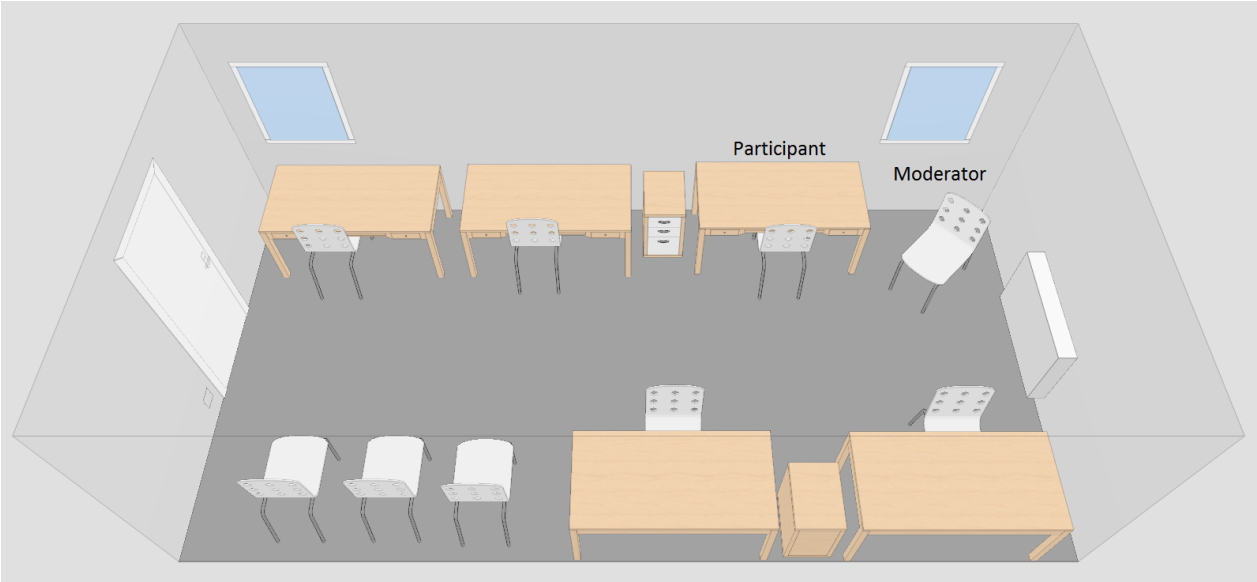


Figure 3.2: Test Setup. Made with IKEA Home Planner

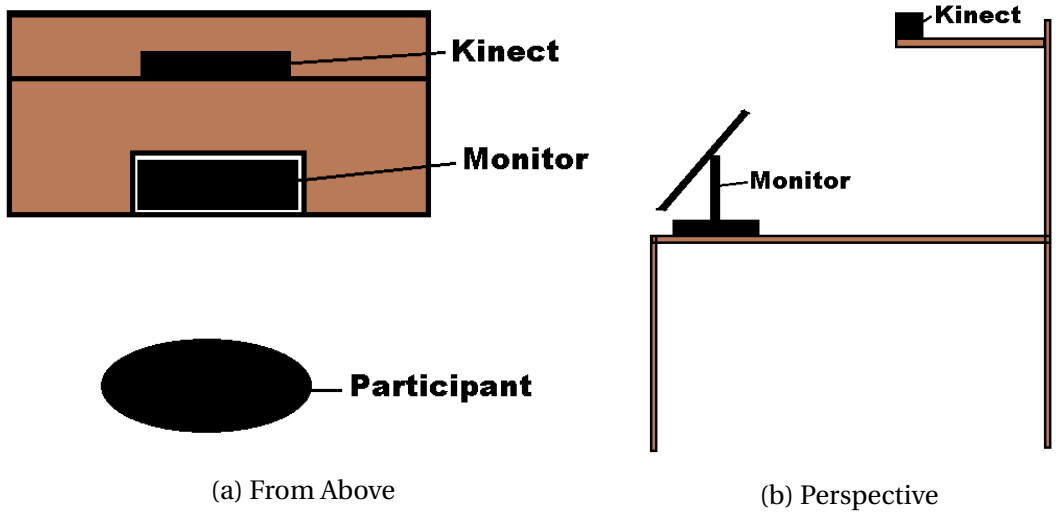
vent the participant from losing focus during the testing. The preparation was aimed at making the interactions feel as natural as possible, similar to how the user would act when performing tasks in a daily work situation. In addition to the test moderator and participant, the desks in the room were occupied by other people. They were working by themselves and were told not to talk. Again, this was to prevent loss of focus.

3.1.3 Test Setup

The hardware used for testing is detailed in table 3.1, excluding the Kinect sensor. The computer was a Dell OptiPlex 9010, chosen because it had hardware sufficient to run both Petrel and the Kinect sensor during the testing. It was positioned as shown in figure XX for the mouse test. The user was seated in front of the computer as one would expect someone to sit and work in an office. The Kinect part of the test was a bit harder. While the sensor is able to interpret gestures from a seated person, it is often more accurate when it is able to see the whole person. The participants were therefore positioned approximately 1.5 meters from the sensor, with the screen and Kinect positioned as demonstrated in figure 3.3. The monitor was tilted approximately 45 deg. The numbers are approximations due to the fact that users would move around during testing, sometimes tilting the screen to suit their height or position. The only important factor was that they remained far enough away so that the Kinect could interpret their gestures

Type	***Name***
Operating System	*Windows 8.1 Pro 64-bit*
CPU	*Intel Core i7-3770*
GPU	*AMD Radeon HD6450 (ATI-102-C26405(B))*
RAM	*16 GB DDR3 SDRAM*
HDD	*Seagate Barracuda ST1000DM003-9YN162*
Screen	*24" Dell UltraSharp U2412M*
Mouse	*Dell M-UAV-DEL8*
Keyboard	*Dell KB212-B*

Table 3.1: Test Computer Specifications



(a) From Above (b) Perspective

Figure 3.3: Test Setup

correctly.

3.2 The User Test

The test aimed to uncover the Kinect's potential as an interaction device for Petrel as compared to the mouse. The tasks were thus made to uncover relative advantages and disadvantages of the devices, both when looked at as a whole and for specific tasks. Is there any area of use where the Kinect might challenge the mouse as the best interaction device in regards to effectiveness, efficiency and user satisfaction? This is the question the test should be able to give some insight into. To create a test that reflects an actual usage scenario, the author of this thesis partici-

pated in a week-long training course about Petrel Geophysics. Participants received a copy of the training manual (Schlumberger (2014)). The test tasks are based on tasks from this manual. This is because the manual was made to get people familiar with Petrel on a basic level, which was useful in our test with inexperienced participants. All test tasks are given in appendix B.

Consisting of three main parts, each covering a different area of use, the test was first completed with the mouse setup and afterwards with the Kinect sensor. This gave the participants experience with the tasks using a familiar interaction device first. The participants could then focus on the Kinect interaction for the second test run, and not on the tasks themselves. The list below shows the general test layout:

- Gain Familiarity
- Data Optimization
- Data Visualization

3.2.1 Task Explanation

For the first part, Gain Familiarity, users were asked to open a 3D window and toggle on 3D seismic data. The user then had to rotate and zoom the object to a certain position, to become familiar with basic menu and window navigation, as this is the foundation for the rest of the test. During the first test run with mouse interactions, the user gains familiarity with the actual program. The task should be more familiar during the second test run with Kinect, so that the focus could be on interacting with the Kinect sensor, and not familiarizing with Petrel. Users were given a thorough introduction to Kinect gestures before beginning.

Figure 3.4, 3.5 and 3.6 show Petrel during the first, second and third test parts respectively. It might be difficult to see on a printed version of this thesis, but the input pane on the left side is what separates the pictures from part one and two. Part one only has the SeismicTestData, while part two has a SeismicTestData [Crop] 1 and a SeismicTestData [Crop] 1 [Realized] 1. This is because part two, Data Optimization, is about taking the seismic volume and optimizing it for

analysis. The model used for testing is small enough that the optimization does not make a significant impact, but the process of optimizing uses 3D window and menu navigation in a way that is intuitive to understand for anyone unfamiliar with geophysics and Petrel. The process was also thoroughly detailed in module 3 of Schlumberger (2014), which made the task creation straight forward. As the test is aimed towards testing program interactions, this was perfect.

The optimization in task two can be broken into three parts, shown in the list below. The cropping reduces the 3D volume's area of focus and computing time. Realizing the volume creates it in ZGY format, which can take time, but is in return much faster than the standard SEG-Y format. Finally, prefetching the volume to cache makes response times even faster. These operations together are useful when working with large data, both to make interactions faster and narrow down the area of focus by cropping.

- Crop
- Realize
- Prefetch

Finally, the 3D window in 3.6 shows the realized and cropped version of the volume from prior test parts, but with the color table set to *Seismic dip azimuth*. The third test part is loosely based on module 4 of Schlumberger (2014) about data visualization. The participant is required to locate the large green area shown on the figure and create a polyline. It should be noted that while the work flow is realistic, the green area of the figure is simply an area where there is no data available. It was used as a focus for this test part because the area is easy to identify for an inexperienced user. Whether or not it is somewhere you would actually create a polyline is unimportant in this test.

3.2.2 Test Design Goals

Task Realism

One of the main design goals was to make the test parts as close to realistic Petrel work flows as possible. This was challenging, given the decision to use students with little to none experience

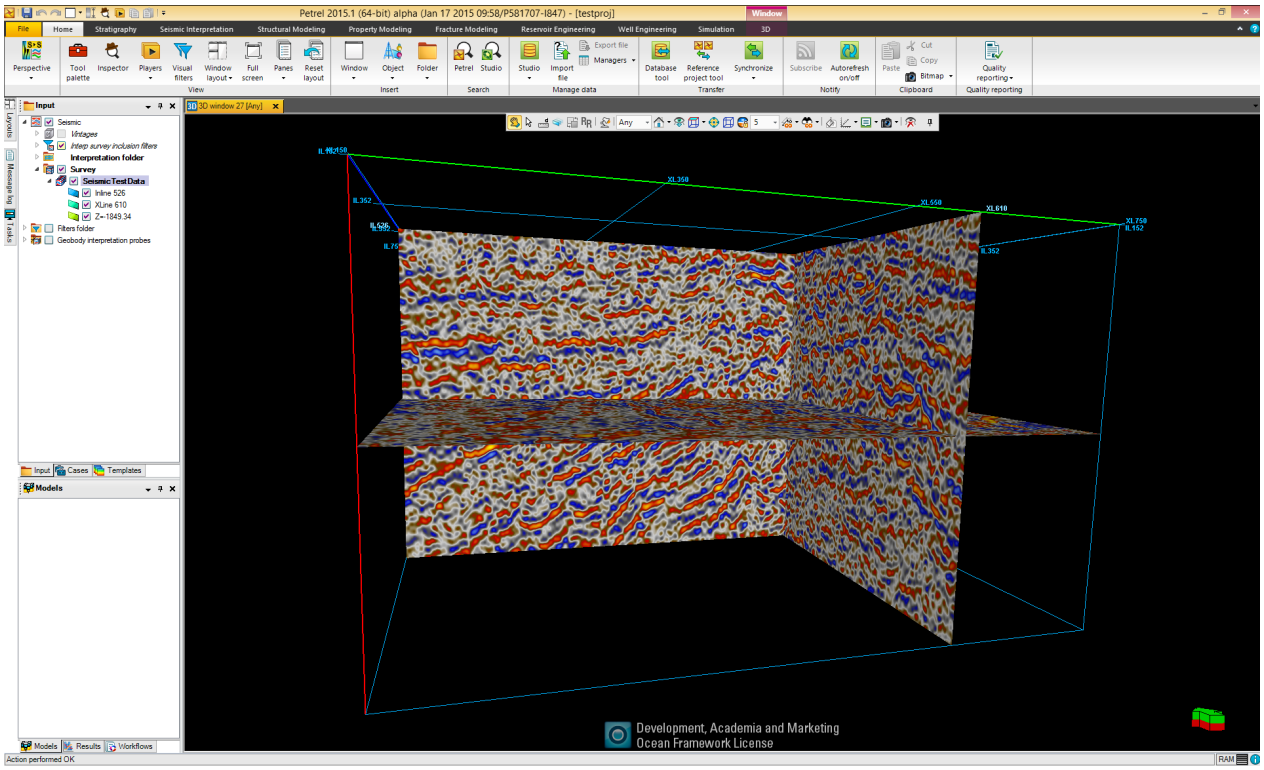


Figure 3.4: Test part one

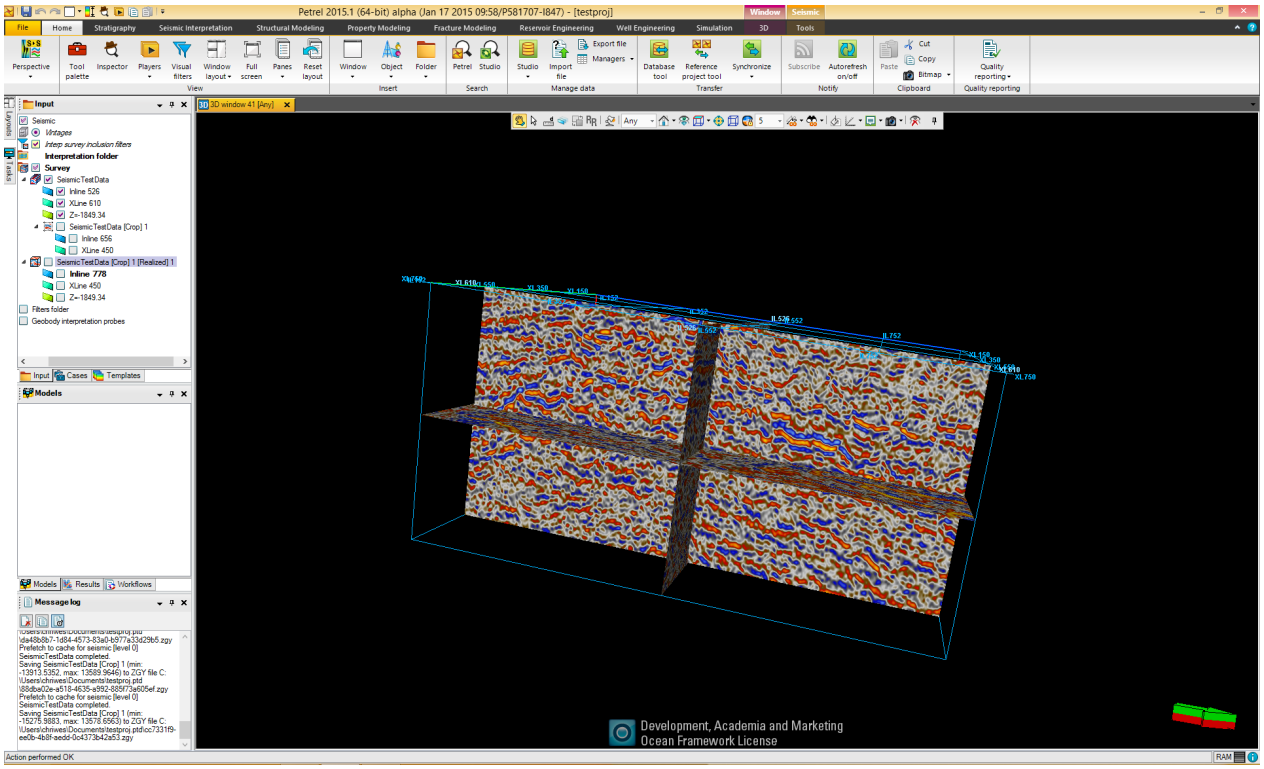


Figure 3.5: Test part two

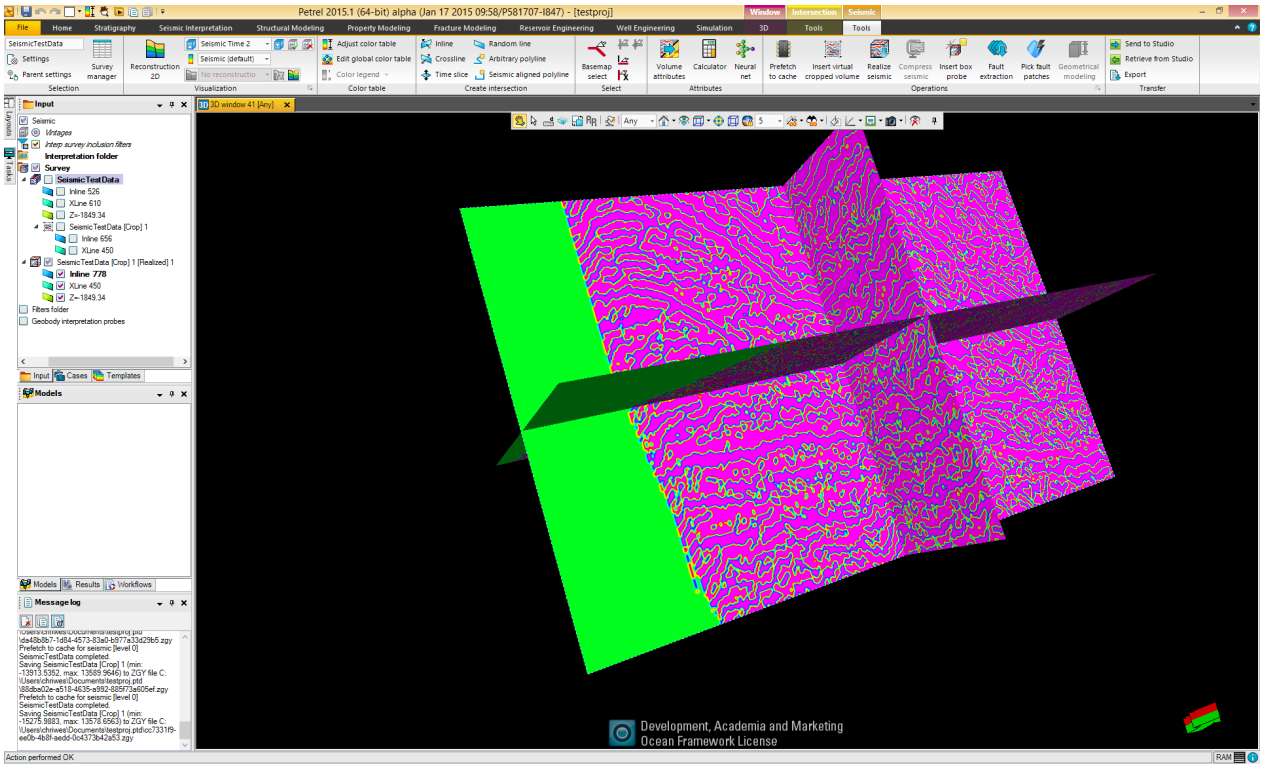


Figure 3.6: Test part three

with Petrel as test participants. As a general rule the different test parts should also have their own areas of focus, so that different aspects of the interaction could be analyzed isolated from each other. This was only achieved to a certain extent, as both menu and 3D window interactions were needed to make the work flow realistic. As mentioned in 3.2.1, the tasks were inspired by assignments in Schlumberger (2014).

Interaction Precision

Another consideration was the different precision levels that an interaction can have. Doing big arm movements to navigate menus and 3D windows was determined to require less precision from the participant than selecting a single point on a 3D volume, or toggling a small button on a menu bar. Between these two levels of precision was assumed to be the zooming and navigation, where the user would hold and drag an object to navigate it's surface. The tasks were made to extract specific information about how the Kinect held up in all of these levels of precision. It is important to note that the precision offered by the Kinect is dependent on the implementation of hand gesture recognitions as well as physical capabilities. So even if the results were to

show imprecisions, this could potentially be remedied by a more detailed implementation. The implemented gestures differ from the previous project (Christopher Benjamin Westlye (2014)), as discussed in the next chapter.

Moderator Intervention

It was expected that the participants would be able to perform the test more or less without moderator intervention. They were still allowed to ask if anything was unclear during the test, but the moderator was required to not initiate contact.

Task Ordering

The mouse part was performed first to let the users become familiar with the tasks. This is so the focus of the test is not put on the actual tasks performed, but rather the interaction used to perform them.

3.3 The Evaluation Method

The evaluation method used is similar to the one used by Alabbasi (2013), whom also collaborated with Schlumberger. It was also used in the previous project (Christopher Benjamin Westlye (2014)) with good results. The method seemed to provide a good initial representation of usability potential. This section describes the different aspects of this approach. All questionnaires used are given in appendix A.

3.3.1 Before the Test

Before the test was handed out, a questionnaire with two parts was filled out by the participant.

Demographic Information

The participants would start out by providing some simple information about themselves, as shown below. The main reason for this was to be able to categorize the participants and potentially find correlations between traits and test feedback afterwards.

Another reason for collecting this data was to increase the time spent interacting with the participant before the test. Further discussed in section 3.3.2, our evaluation approach was to have the participant share as much as possible during the test. It was therefore important to make the participant feel relaxed beforehand, so that they would be more willing to share their thoughts.

- Name
- Gender
- Age
- Educational Background
- Current Position

Experience Information

Since the test is inherently technical, revealing the participant's capabilities and experiences is useful when comparing the results. The question genres are shown below. The participants were first asked about their daily computer usage, as this could have relevance to the mouse part results. They were then asked about how often they use software that included some sort of tasks. This includes all tools where the goal is to perform actions on data objects. Finally, the participants were asked about any previous experience with the Kinect. Since the Kinect is different from most kinds of interaction devices, prior experience could significantly affect the test results.

- Computer Use
- Software based around tasks
- Petrel
- Kinect

3.3.2 During the Test

The Think Aloud Method

The Think Aloud Method was used during testing. Described in van Someren et al. (1994), it revolves around making the participant vocalize their experiences while experiencing them. This gives an insight into the thought process of the participant, and provides valuable feedback on how interactions with the presented tools are actually happening. A person may feel differently about the interaction afterwards when they have processed the information, so this approach allows us to analyze how the person felt when performing the test.

Users were encouraged to be vocal about their thinking through the whole testing process. Whenever the participant was too quiet, the moderator asked questions to reengage the participant vocally. This dialog was recorded for analysis in the report to prevent the moderator from forgetting information.

3.3.3 After Each Test Part

To be able to get as much information about the user experience as possible, some kind of feedback form was needed between the mouse and Kinect parts of the testing. There exists many different approaches to gaining feedback after a test. One approach, discussed below, stood out when it came to simplicity and feedback quality.

The System Usability Scale (SUS)

Developed in 1986 and described in Brooke (1996), the SU scale consists of ten Likert-scale questions. The user gives an answer between 1: strongly disagree and 2: strongly agree after being presented with various subjective statements about the interface. This scale was chosen because it provides high-level feedback about the system. The questions on the SUS cover usability aspects related to effectiveness, efficiency and user satisfaction. These are mentioned as suitable usability measures in International Organization For Standardization (1998).

To calculate a total score for the SUS answers, all odd items have one subtracted from them.

The even items are subtracted from five. This causes all answers to be on the scale 0-4 with zero being the lowest and four the highest possible score, as higher score on even items originally means lower score. The resulting numbers are added and multiplied by 2.5 to scale scoring range to 0-100. The average SUS score is 68, so anything below this is considered below average.

3.3.4 After The Test

In addition to the SUS questionnaire after each part, the participants were asked three questions after the test. The subjects for these are described below. The purpose was to figure out whether one interaction method stood out to the participants with regards to the given aspects. The questions are provided in appendix A.

Effectiveness

The first question was about which interaction method that worked the best with regards to effectiveness. Effectiveness here encompassing the accuracy and completeness with which the participant could achieve their task goals.

Efficiency

The next question was related to how much effort the participant had to expend to achieve desired results. This included both mental and physical work, as the participants had to exert themselves physically when interacting with the Kinect.

User Satisfaction

The last question asked which tool the participant was the most satisfied with. International Organization For Standardization (1998) mentions typical unsatisfying elements as to whether or not the user was free from discomfort and perception of the interaction tool that the user is left with afterwards. The results for the questions about effectiveness, efficiency and satisfaction can be found in 5.3.1.

3.4 The Test Participants

3.4.1 Number of Participants

It was important to have enough participants so that the results would be credible. Existing literature, notably Nielsen (1993) and Virzi (1992) argue that five users is enough to identify the most common usability issues. This proved sufficient for previous testing. It should be noted however, that this number is disputed, for example by Faulkner (2003), especially when it comes to identifying the more uncommon usability problems. The original plan was therefore to use 20 participants in this project. After the testing started, it became apparent that feedback from each participant was very similar. When number of participants came close to ten, little new information was obtained. The testing thus ended with ten participants.

3.4.2 Considerations when Selecting Participants

Since the study was about interaction tools in an office setting, it seemed natural to use people that worked or would work in such an environment. Due to easy access, the participants were chosen to be students at any MSc program. The reasoning was that people with little work experience, but sufficient technical expertise would be more open towards trying new interaction devices. John Lester Sarmiento Gerardo (2007) discusses that novice users are not only sufficient, but can provide advantages over experts by uncovering more errors.

Chapter 4

Test Implementation

This chapter details the program implementation used in the usability study. Based on test results from the project thesis (Christopher Benjamin Westlye (2014)), Kinect support was implemented for Petrel. Section 4.1 talks about the code design. Initially, the implementation was a plugin using Petrel's API, Ocean. A better solution became apparent however, making the final implementation very different from the initial prototype. It is not discussed here since it was never used, even though the development used up much of the project time. Sections 4.2 and 4.3 present the interaction design for both the testing interfaces.

4.1 Code Design

Using the Ocean SDK required an implementation of 3D window interaction from scratch. It therefore made more sense to simply feed Kinect point data directly to the screen pointer, so that the interaction with mouse and Kinect were as similar as possible. Making the Kinect control the pointer directly also wouldn't artificially limit the user to certain parts of Petrel. It would instead feel like a true alternative to mouse interaction. The overall code structure can be seen in figure 4.1, with each class explained below.

4.1.1 Main Window

As in the earlier project, the code was written mainly in C# with some XAML. The MainWindow class, shown in 4.2, handles logic for the application's GUI. The different methods are executed

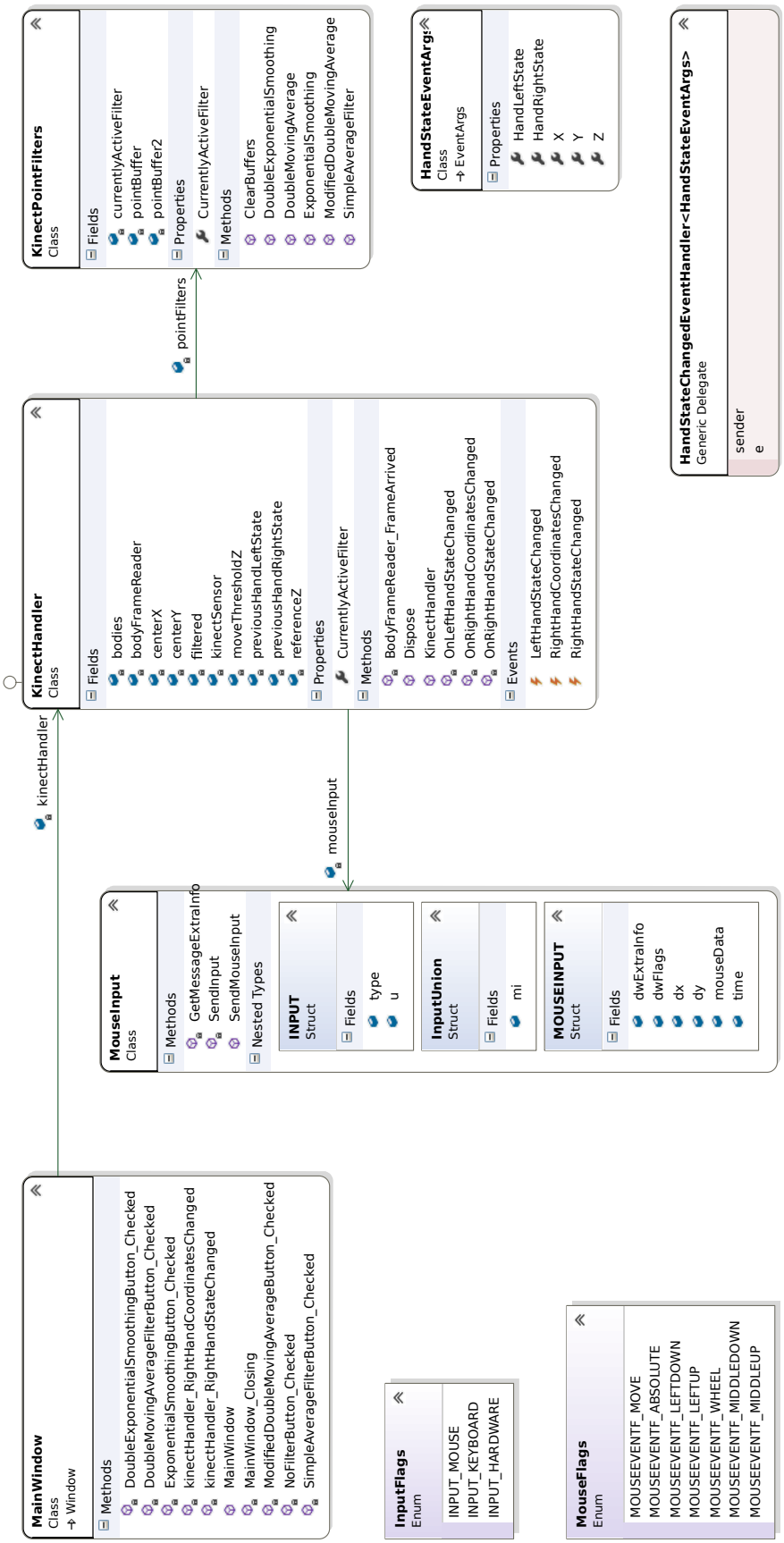


Figure 4.1: Application class diagram

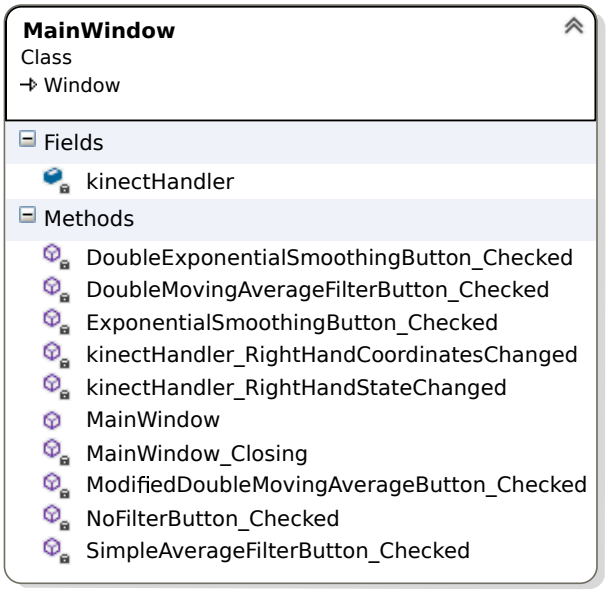


Figure 4.2: MainWindow class

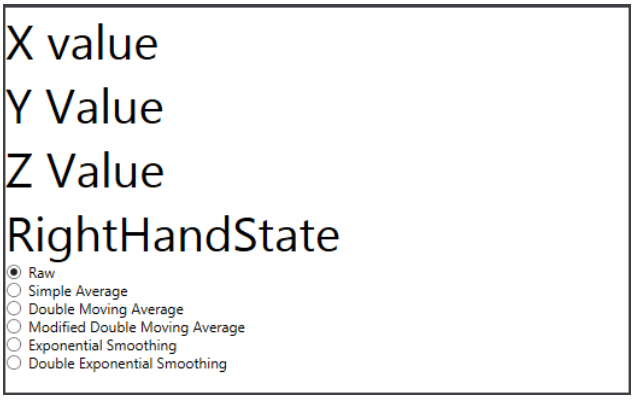


Figure 4.3: Test Application UI

upon checking different options during runtime. The UI is shown in figure 4.3. It shows some information about hand state and current filters and is never seen by test participants. This class is also the first to be initialized, and continues by initializing the Kinect Handler which is the most important part of the program.

4.1.2 Kinect Handler

The KinectHandler class detailed in figure 4.4 receives and uses Kinect input. When initialized, it subscribes it's method BodyFrameReader_FrameArrived to the Kinect event that triggers when-

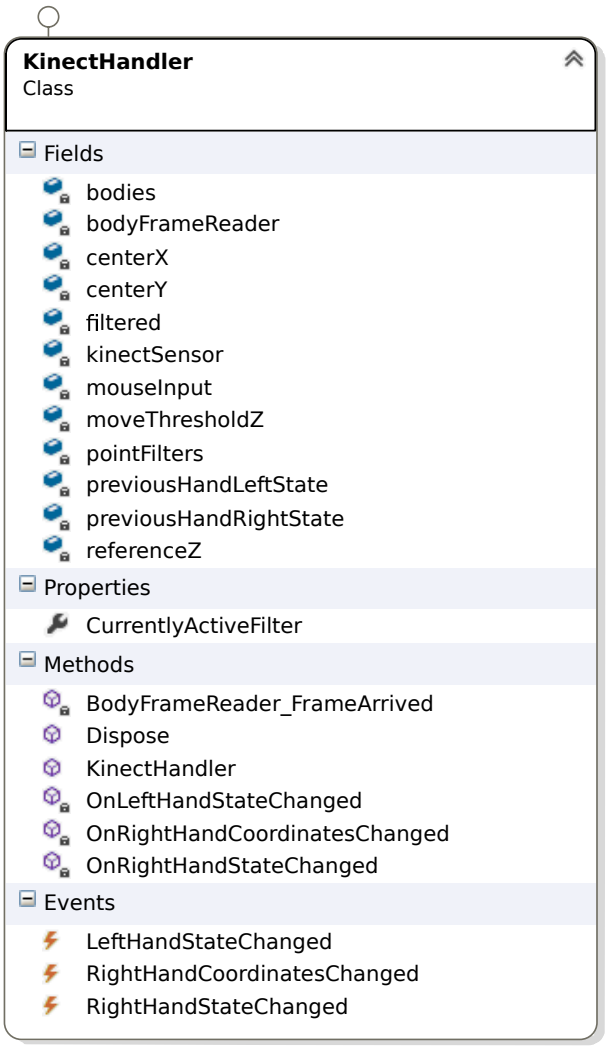


Figure 4.4: KinectHandler class

ever a body is detected. Information from this body is then stored in the class fields. Position and state of hands are retrieved and forwarded to the MouseInput object. The position data is filtered if this is selected in the GUI.

4.1.3 Kinect Point Filters

Hand position data from the Kinect was at first fed directly to the screen cursor. This proved to be inaccurate, since the input was not free from noise. Several different filters were implemented to remedy this situation, all of them listed below. The simple average method was the first to be implemented. It averaged the five latest points whenever a new point was retrieved

from the Kinect. The double moving average smoothing averaged five points each time a new position was received just like the simple average, but then proceeded to put the averaged point into a second buffer. When the second buffer reached five points they were then averaged again, and the resulting point was fed to the screen pointer. This caused outliers to have less of an impact, but in return caused a small delay in response time compared to not using any filtering. A modified version was implemented to increase response times, but still had some noise issues.

Implemented filters:

- Simple Average Smoothing
- Double Moving Average Smoothing
- Modified Double Moving Average Smoothing
- Exponential Smoothing
- Double Exponential Smoothing

The three smoothing methods explained above made the Kinect interaction smoother than with no filter as was used in Christopher Benjamin Westlye (2014). However it still did not feel as responsive and smooth as computer mouse interaction. It was not until exponential smoothing was implemented that the desired interaction experience was obtained. This was determined through simply trying out the software, and all smoothing methods are included in the code attachment for this thesis. Simple exponential smoothing was implemented, adding a weight α to the newest hand position and a weight $1-\alpha$ to the previous point. alpha was decided through trial and error, with $\alpha = 0.4$ providing the best interaction experience. The resulting point after running exponential smoothing was still slightly noisy, so double exponential smoothing ended up being used for the usability test. This method is similar to exponential smoothing, except that trend smoothing is incorporated, causing a smoother response whenever there is a trend present in the data. Diagram for the class implementing the filters can be seen in figure 4.5.

For t = 1:

$$s_1 = p_1 \tag{4.1}$$

$$b_1 = p_1 - p_0 \tag{4.2}$$

For t > 1:

$$s_t = \alpha p_t + (1-\alpha)(s_{t-1} + b_{t-1}) \tag{4.3}$$

$$b_t = \beta(s_t - s_{t-1}) + (1-\beta) b_{t-1} \tag{4.4}$$

The formulas are given by 4.1, 4.2, 4.3 and 4.4. Starting at $t = 0$, p_t is the hand position at time t, s_t is the smoothed hand position at time t and b_t is an estimate of the trend at time t. α is referred to as the *data smoothing factor*, $0 < \alpha < 1$, set to 0.4 for the testing. This emphasizes earlier values slightly more than present ones, causing the mouse cursor to exhibit smooth movements. The reasoning is that if a new position input is noisy and does not correlate with actual hand movement, it has less of an impact than all the position inputs that are accurate and thus should be in the same area. In short, outliers are smoothed out.

β is the *trend smoothing factor*, $0 < \beta < 1$, set to 0.5 for the user test. Thus, in our case equation 4.4 becomes equation 4.5, averaging the current and previous trend estimates. The hand movements required to move the cursor follow a trend, especially when seen from the Kinect's sampling perspective. The addition of trend recognition thus helps smooth out screen cursor movement.

$$b_t = \beta(s_t - s_{t-1}) + (1-\beta) b_{t-1} = \frac{(s_t - s_{t-1}) + b_{t-1}}{2} \tag{4.5}$$

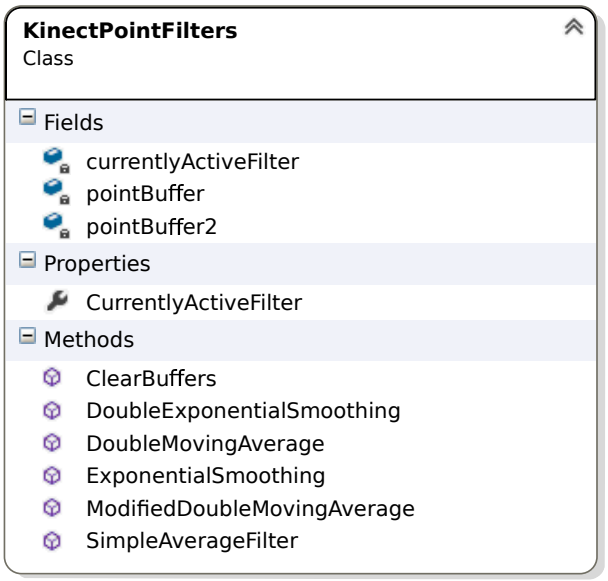


Figure 4.5: KinectPointFilters class

```
[DllImport("user32.dll", SetLastError = true)]
static extern uint SendInput(uint nInputs, INPUT[] pInputs, int cbSize);
```

Figure 4.6: SendInput method

4.1.4 Mouse Input

The title is somewhat misleading, as mouse input in this case refers to inputting the Kinect positional data to the screen cursor. The class was called MouseInput due to the inherent relationship between the screen cursor and the computer mouse. Figure 4.6 shows the Windows API method used to control the screen cursor. A diagram for the class responsible for this is shown in figure 4.7. The main responsibility of this class is to import and abstract away the low-level SendInput method.

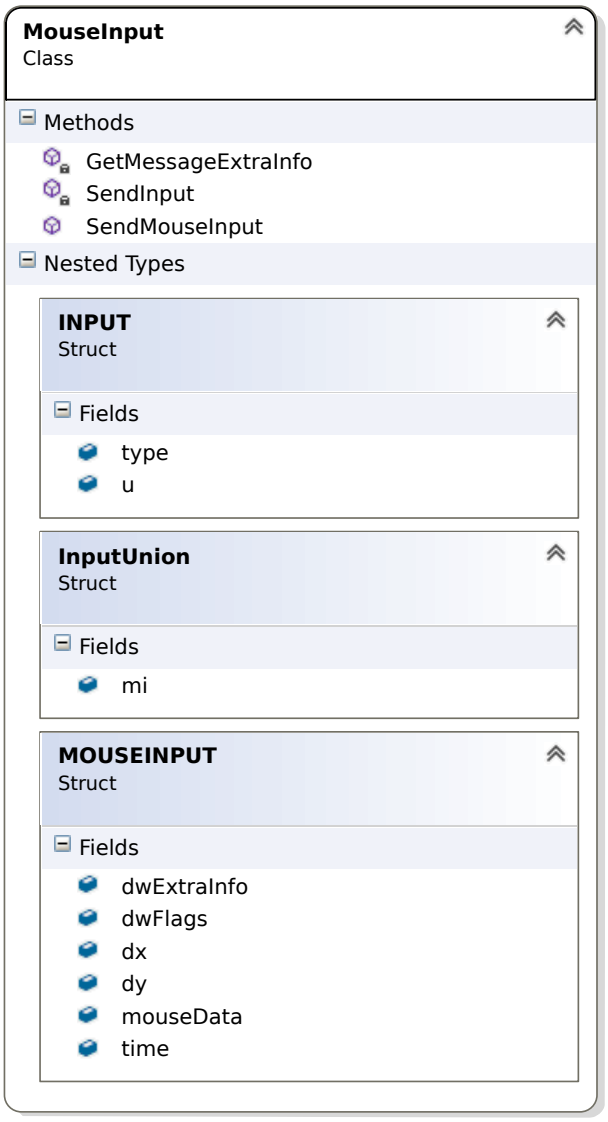


Figure 4.7: MouseInput class

4.2 Mouse Interaction

Before getting into the different implemented Kinect gestures, the equivalent resulting operations are presented with regards to the mouse. To avoid unnecessary complications, this thesis does not consider the use of a keyboard. All implemented interactions are therefore with either a mouse or the Kinect. This can be clearly seen in the test tasks presented in appendix B, where only the mouse actions shown in the list below are required by the participant.

- Mouse movement
- Left mouse button click
- Left mouse button hold
- Scrolling wheel forward roll
- Scrolling wheel backward roll

The test mainly requires interaction with menus, windows and the 3D seismic model. The menu and window interactions are not explained further with regards to the mouse, as left clicking to navigate these should be known by most people. The 3D seismic interaction is explained below.

4.2.1 3D seismic interaction

Figure 4.8 shows the seismic model used in the test from the side. To interact with it using the mouse, the view option on the window toolbar has to be toggled on. If this is done, left mouse button hold combined with mouse movement will rotate the model. This interaction is a native part of Petrel. Doing this and releasing the left mouse button after a movement will cause the model to spin around until manually stopped. This is the reason left mouse button click is a separate entry, as clicking or resuming the left button hold will cancel this spin. This becomes more important when the Kinect interactions are introduced in 4.3. Rolling the scrolling wheel forward will zoom in on the model and rolling the wheel backwards will zoom out. This sums up the most important part of the test interaction.

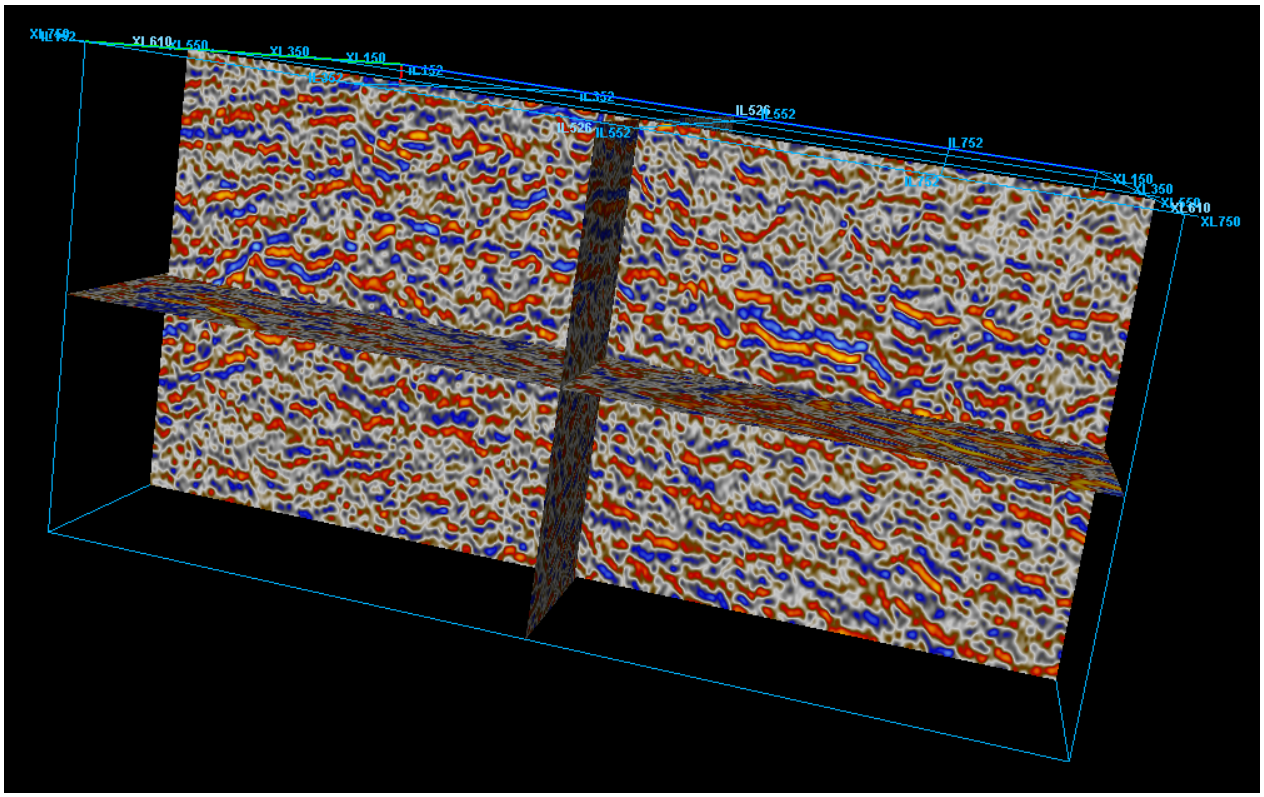


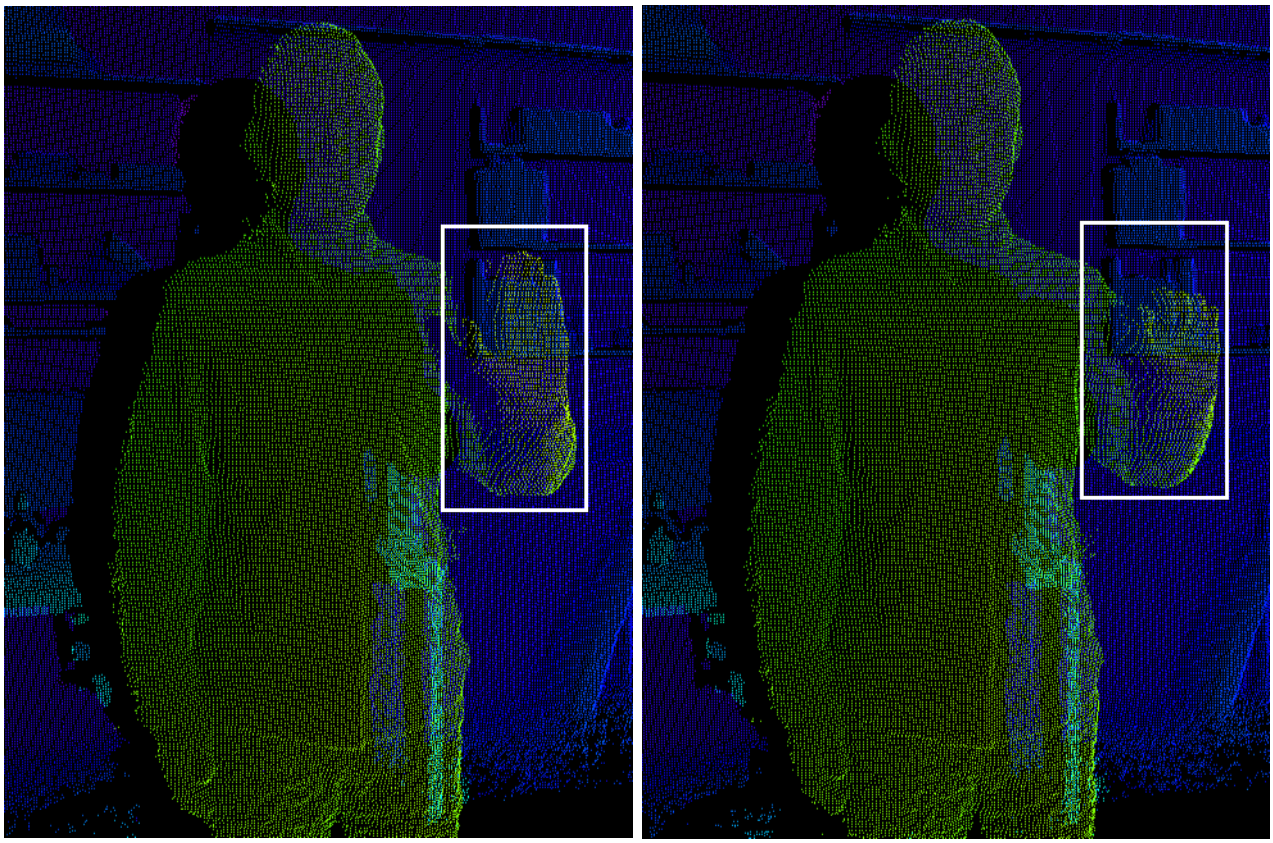
Figure 4.8: Petrel 3D seismic

4.3 Kinect Interaction

The Kinect gestures supported for this usability study are a result of experience from the previous project (Christopher Benjamin Westlye (2014)) and analysis of existing literature, especially the Microsoft's published guidelines (Microsoft (2014)). Each individual interaction is explained in detail below with illustrations. Keep in mind that the images in this section are mirror images.

4.3.1 Moving and Holding

Using the right hand for mouse interaction was required from test participants. With this in mind, the right hand was chosen as the main hand for cursor interaction with Kinect. Upon engaging the Kinect, the cursor thus follows the participant's right hand with the center of the screen correlating to the center of the Kinect. To avoid the Kinect from accidentally disengaging the participant during testing, specific gestures for engaging and disengaging Kinect interaction was omitted unlike last time. The Kinect always tracks the participant's right hand. This caused



(a) Right hand open (b) Right hand closed

Figure 4.9: Right hand states

some problems, as described later in this section. To perform the equivalent of holding the left mouse down, the state of the right hand was used. An open hand meant that the left mouse button is not pressed and a closed hand that it should be pressed and held until the hand is open again. Images demonstrating the gestures seen from the Kinect's depth sensor are shown in figure 4.9.

4.3.2 Zooming and Scrolling

While scrolling is of no significance in the usability test, zooming is important when interacting with the 3D seismic model. The lasso hand gesture with the right hand was used for this. It is demonstrated in figure 4.10. Supported by the Kinect, this gesture is performed by pointing the index and middle fingers upwards while the rest of the fingers are closed. The implementation in the KinectHandler class saves right hand position whenever the lasso gesture is detected. While

the lasso gesture is held, the z component of all new hand positions will be compared to the z of the initial position as demonstrated in algorithm 1 below. The result is that holding the hand as shown in 4.11a will continuously zoom in while holding the hand as shown in 4.11b will continuously zoom out. In the last project, users had to hold and drag to zoom, then upon reaching an uncomfortable position restart the gesture to zoom further. The intention was to simulate interaction with a physical object, but proved to just unnecessarily complicate the interaction. This is the reason users this time only have to hold their gesture relative to the reference until desired zooming or scrolling is achieved. Figure 4.12 shows the Kinect's coordinate system.

Data: Right hand state
Result: Mouse scrolling wheel input

if $RightHandState_{current} = lasso$ **then**
  if $RightHandState_{previous} \neq lasso$ **then**
    $Z_{ref} = RightHandPosition.Z$

while $RightHandState_{current} = lasso$ **do**
  if $Z_{ref} - Z \geq threshold$ **then**
    Simulate mouse wheel forward scrolling
  else if $Z_{ref} - Z \leq -threshold$ **then**
    Simulate mouse wheel backward scrolling
  else
    Do nothing

$RightHandState_{previous} \leftarrow RightHandState_{current}$;

Algorithm 1: Kinect's equivalent to mouse wheel scrolling

4.3.3 Precision Clicking

The previous implementation lacked precision in its gesture for performing the equivalent of a left mouse click. The solution is shown in figure 4.13. Users let the screen cursor track their right hand as usual, but instead of clicking by closing the right hand, users close their left hand. This considerably improved precision compared to last time as the left hand has nothing to do with position inputs. By only registering the hand state, users can use the Kinect in a very precise way. While an intuitive one handed gesture with the same levels of precision would be ideal, the two-hand approach provides sufficient precision for test execution.

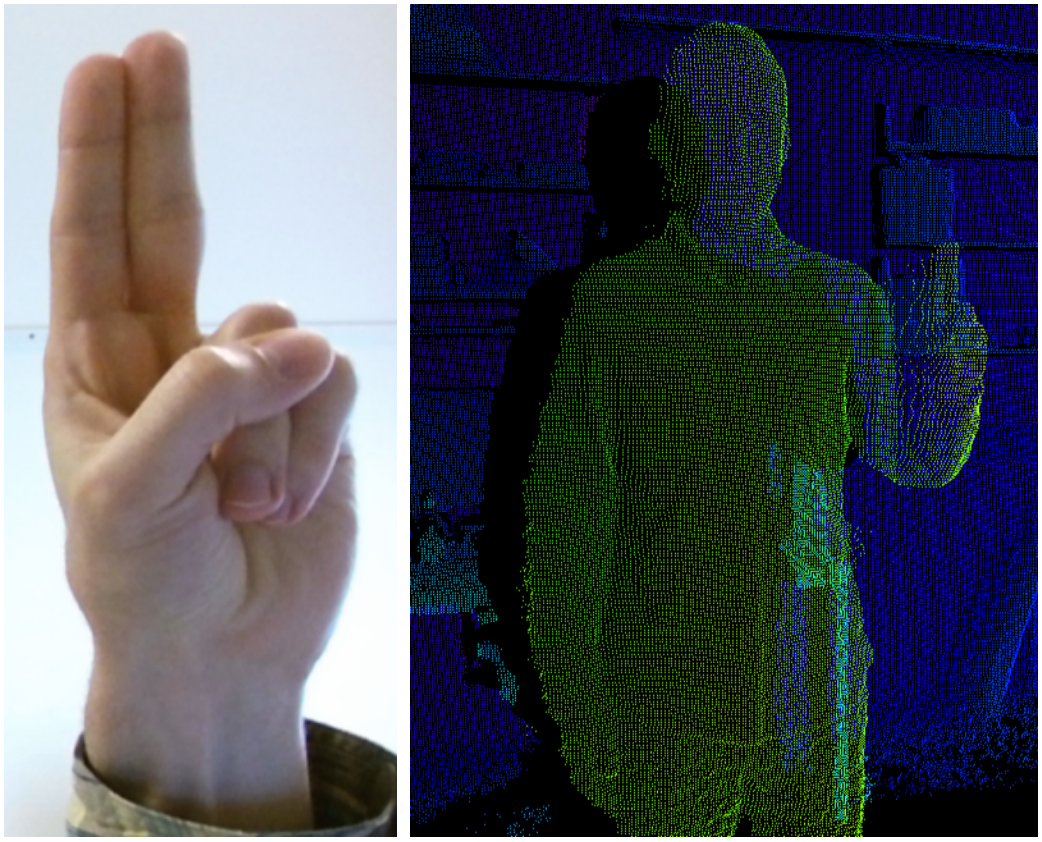
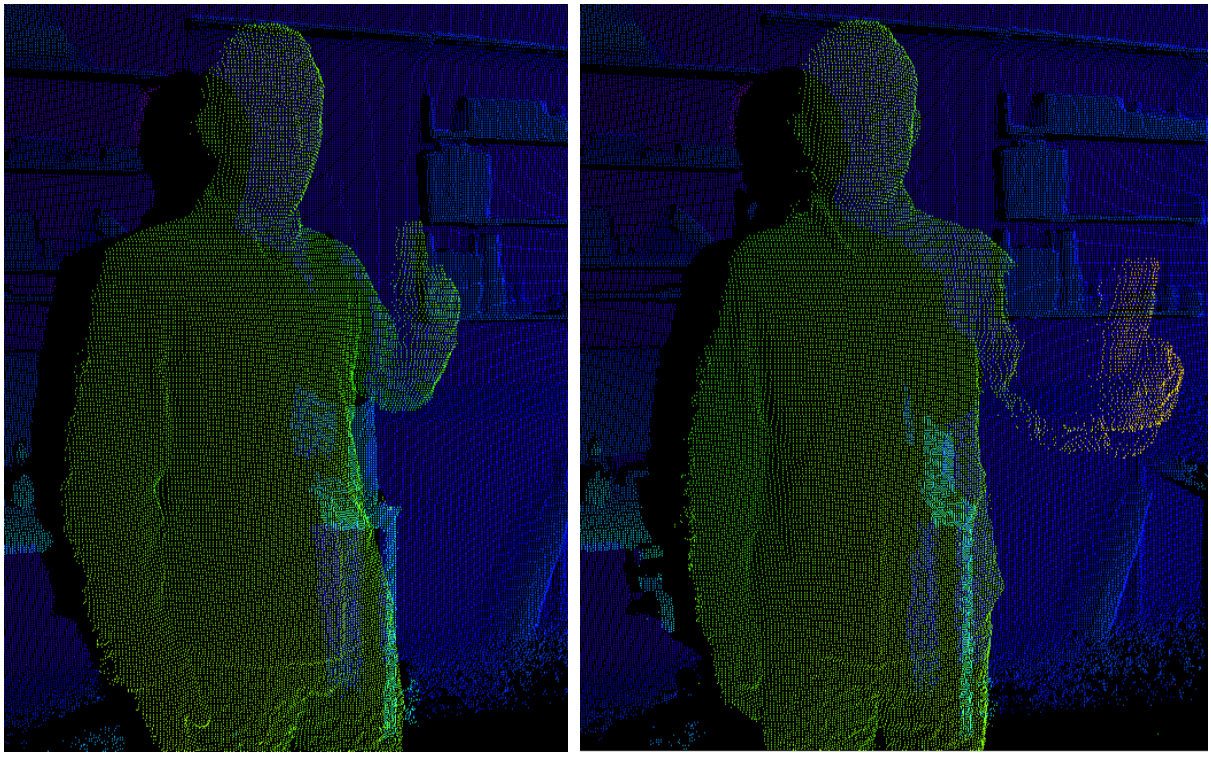


Figure 4.10: Right hand lasso



(a) Right hand lasso zoom in (b) Right hand lasso zoom out

Figure 4.11: Lasso hand movement along z-axis

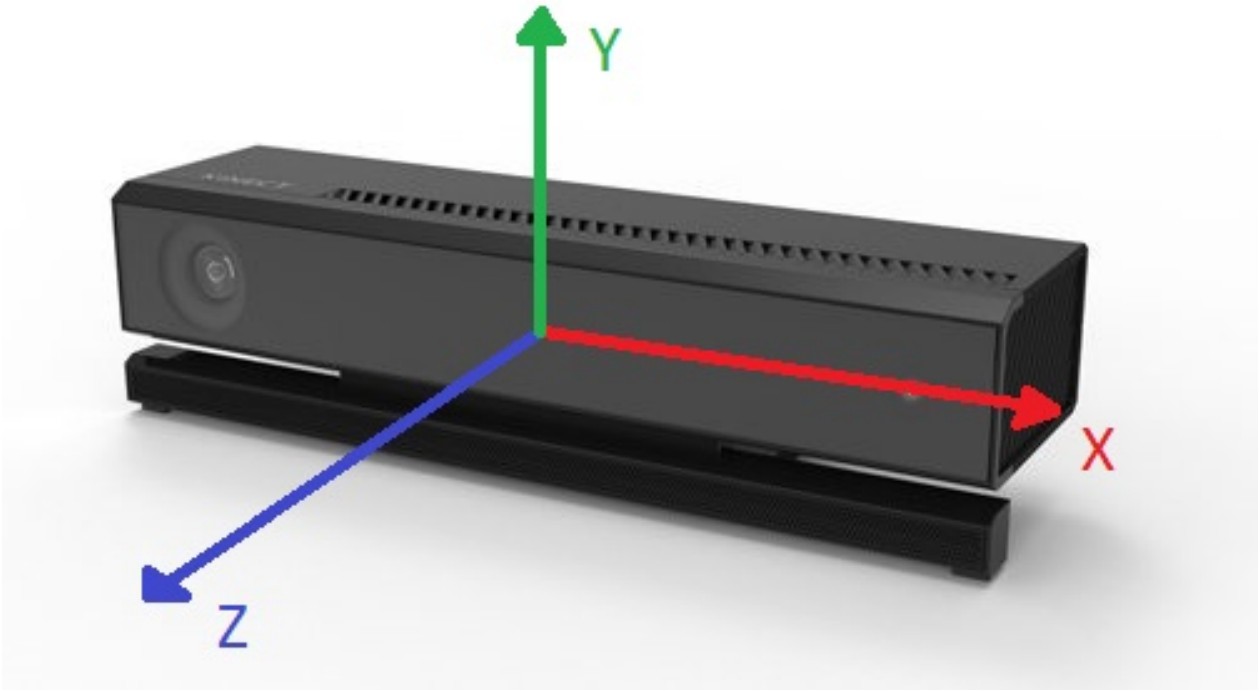


Figure 4.12: Kinect coordinate system

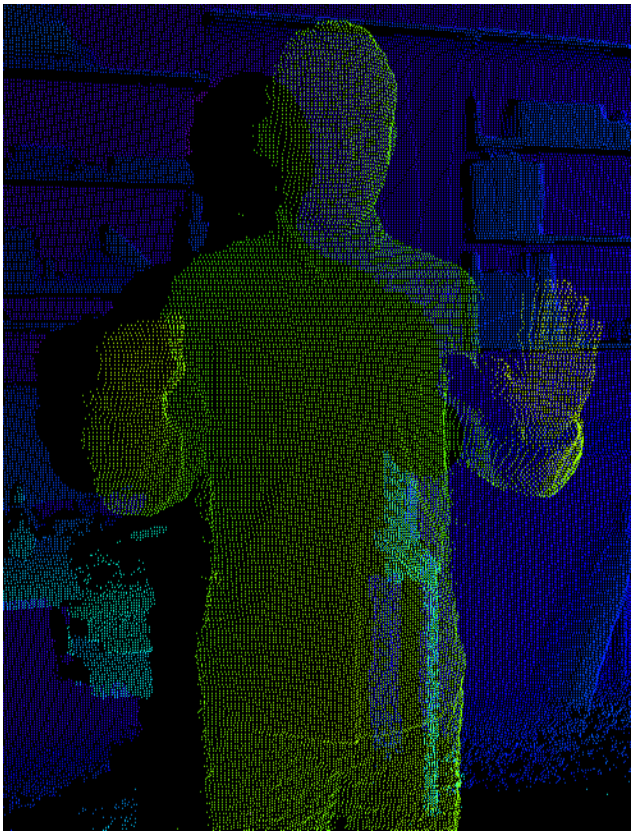
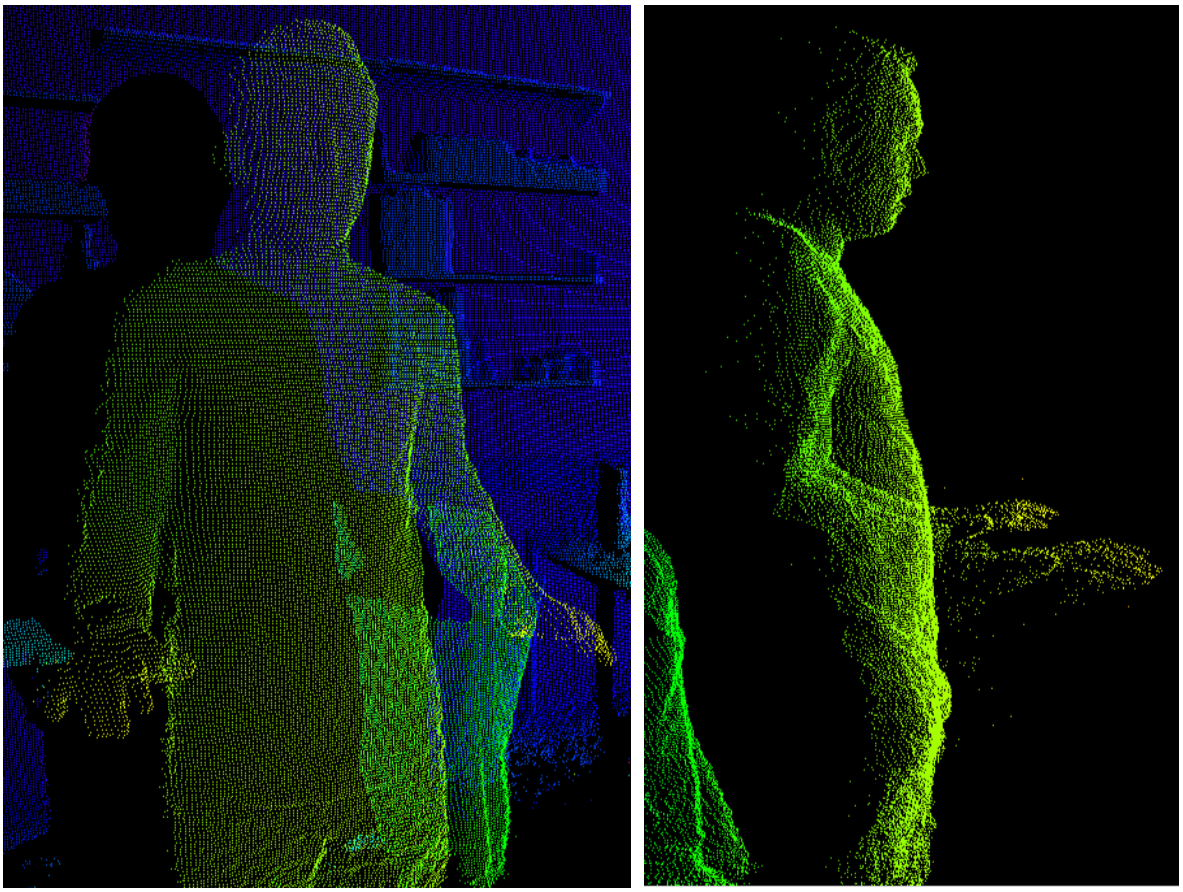


Figure 4.13: Left hand closed

Figure 4.14: Engaging the Kinect. Figure from Microsoft (2014)

4.3.4 False Positives

To prevent the Kinect from unintentionally disengaging its user in the middle of a gesture and removing their feeling of immersion, specific gestures for engaging and disengaging the Kinect were not included in the implementation unlike in previous testing, where users had to hold their hand as shown in figure 4.14 to start the interaction. This design decision lead to both positive and negative effects. It prevents the Kinect from misinterpreting gestures as disengages and thereby breaking immersion. But it also lead to more false positives. Whenever users lower their hands, the Kinect does not necessarily stop detecting gestures. Gestures meant for other people can therefore lead to misclicks. Figure 4.15 shows a person in the act of lowering their hands. From the Kinect's perspective this is sometimes recognized as the person having closed hands. As mentioned in section 4.3.1 this causes a left mouse hold, potentially pressing some unintended button on the screen.



(a) Front (b) Perspective

Figure 4.15: Common Reason for Positive Triggers

Chapter 5

Results and Analysis

This chapter presents and discusses the usability test results. Section 5.1 presents the study participants, how they were selected and their experience with relevant interaction tools. Section 5.2 attempts to interpret the numerical data obtained from the test. Finally, section 5.3 looks at feedback from both during and after the testing in a qualitative setting.

5.1 Demography

5.1.1 Participant Selection

Ten students participated in the study, five of whom also partook in the previous study (Christopher Benjamin Westlye (2014)). As mentioned in earlier chapters, participants were selected to be students from technical or scientific fields. Besides being easily available, they represent a group about to enter the work force, where most will end up working in an office setting. Office use is the target area of use for this usability study, so the selection seemed appropriate. Table 5.1 shows which fields are represented and how many participants were from each field. Figure 5.1 shows the age distribution for participants, ranging from 21 to 29. The sample consisted of two female and eight male students.

Field of Study	Number of Participants
Applied Physics and Mathematics	1
Communication Technology	1
Chemical Engineering and Biotechnology	2
Computer Science	2
Engineering Cybernetics	2
Mechanical Engineering	2

Table 5.1: Participant Data

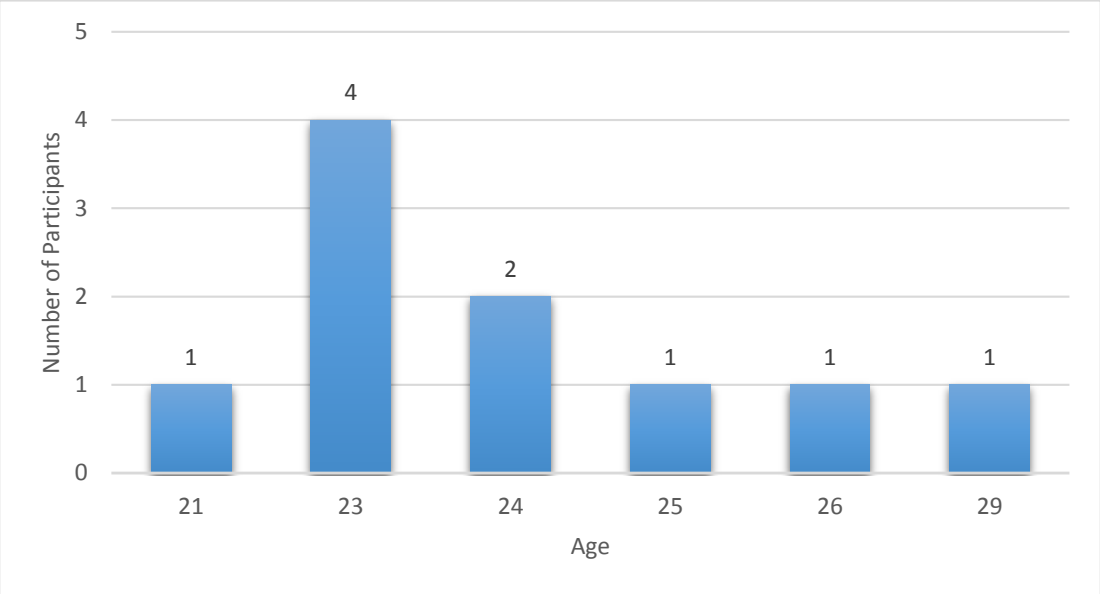


Figure 5.1: Age of Participants

5.1.2 Participant Experience

The same testers as last time were chosen again to include people with Kinect experience, as this was lacking in most people asked to participate. As the pie chart in figure 5.2 shows, the user sample ended up including more students who had used the Kinect before than students who had not. This experience was reported to be from either trying out Xbox games, at demonstrations featuring Kinect or participating in the previous test. Some participants reported experience from several of these. The main impact of this is considered to be a reduced need for an introduction of the Kinect, since users knew what to expect. Previous experience could impact SUS scoring however, as the Kinect becomes easier to use the more one uses it. To reduce this effect, all participants were given a thorough introduction to Kinect gestures and allowed to play around with it until they felt comfortable before commencing testing. This was not the case in the previous test, which resulted in a certain amount of unpredictability in test time usage, as the participants used test time to figure out gestures on their own. The lack of comfort resulting from not knowing gestures could also have impacted SUS scores.

When it came to the Petrel software on the other hand, figure 5.3 shows that only one participant had used Petrel before. This could have huge consequences for the results, as Petrel is considered expert software and has a lot of functionality. Normally, it is only used by people with coursing on its use and an understanding of geophysics. As presented in the previous section, none of the participants were from Petroleum studies. With this in mind, it became very important that the testing required minimal understanding of seismic data and instead focused on pure usability and interaction. This was made easier by having access to the Schlumberger (2014) manual mentioned in section 3.2 which contained well made introductory assignments.

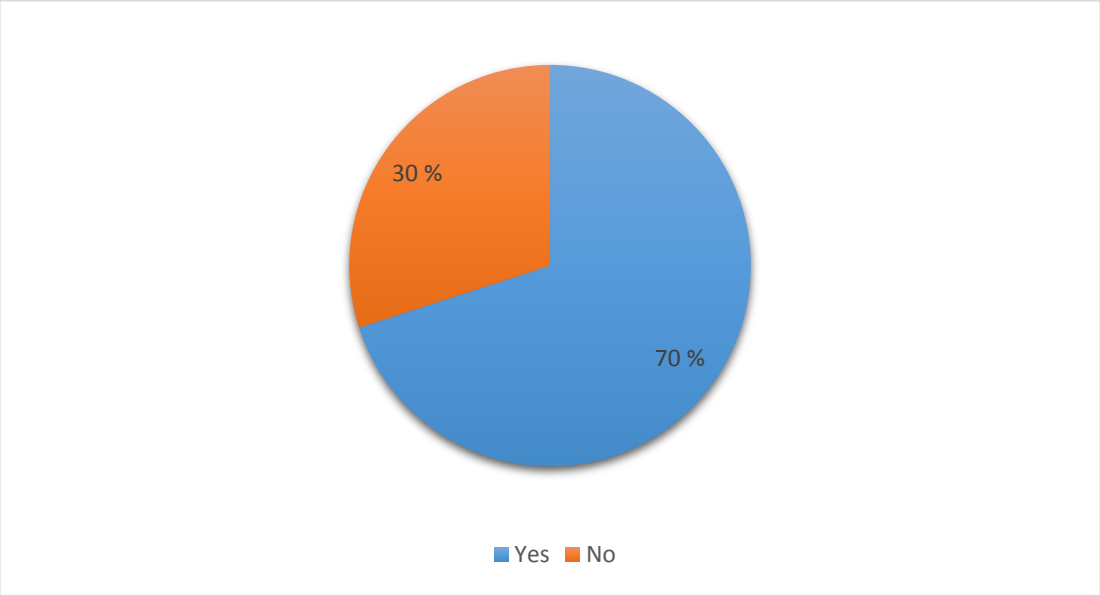


Figure 5.2: Kinect Experience

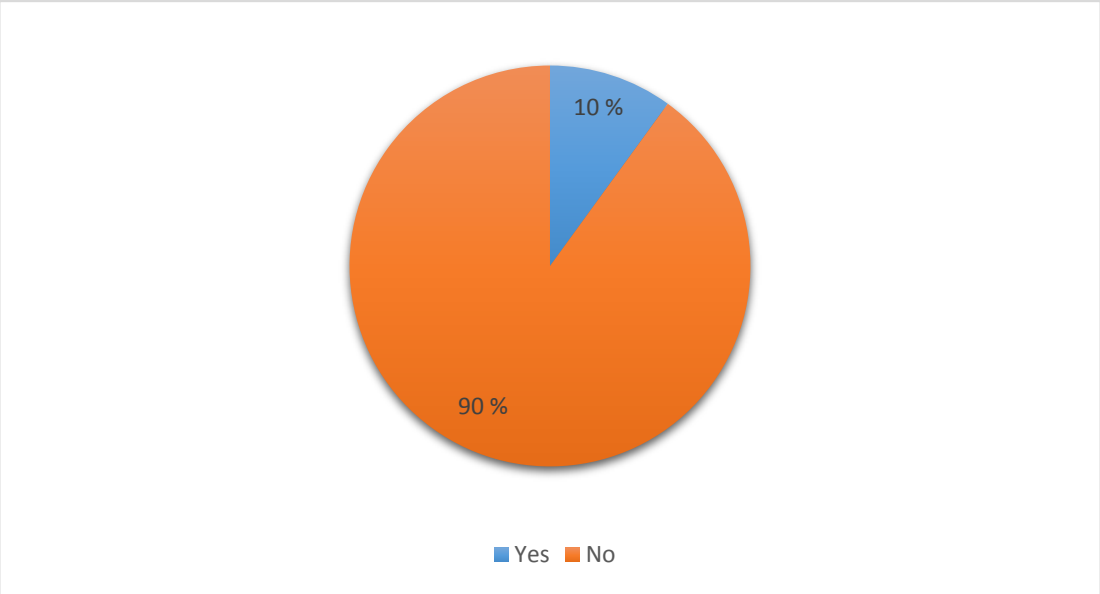


Figure 5.3: Petrel Experience

5.2 Quantitative Results

This section presents and discusses the quantitative test results. SUS scores and time usage are the main forms of quantitative results, so these will be presented at first by themselves and afterwards in relation to various factors. It should be noted that quantitative data from ten participants are limited and not necessarily representative of the population, or even the target subgroup consisting of students with technical experience. The most important data is therefore qualitative data resulting from detailed test observation, the think-aloud method presented in section 3.3.3 and test session recordings. Participants were overwhelmingly helpful in their feedback both during and after testing. Some users commented more than others, but almost everyone contributed their thoughts on potential area of use and gave reasons for their scoring on each individual SUS entry. As discussed in section 3.4, the amount of testers used is sufficient for identifying common usability issues and providing an overview of the relative usefulness of the interaction devices.

5.2.1 SUS Scores

See section 3.3.3 for details on calculating SUS scores. Test results are given in figure 5.4 for the mouse interaction and in figure 5.5 for the Kinect interaction. At first glance it may seem that average scores are pretty close. To make better sense of the data, the rest of this section is dedicated to extracting statistical information and analyzing the results.

Evaluating the Arithmetic Mean

Arithmetic means for SUS scores are given in table 5.2. When looking at all users, the Kinect scores are 2.2% higher on average than mouse scores. Keeping in mind that the user sample is small and scores could be very different if it was bigger, the results tell us a lot about how the interfaces were perceived relative to each other. First of all, the scores for both interfaces are very close. This was not the case with the last test, where the mouse trumped the Kinect by a large margin. Arithmetic means for the previous test were 74 for the mouse and 48 for the Kinect. The Kinect thus scored 35.1% lower than the mouse. The tests can not be directly comparable in all areas as Kinect implementation, test application and user tasks are different. Especially

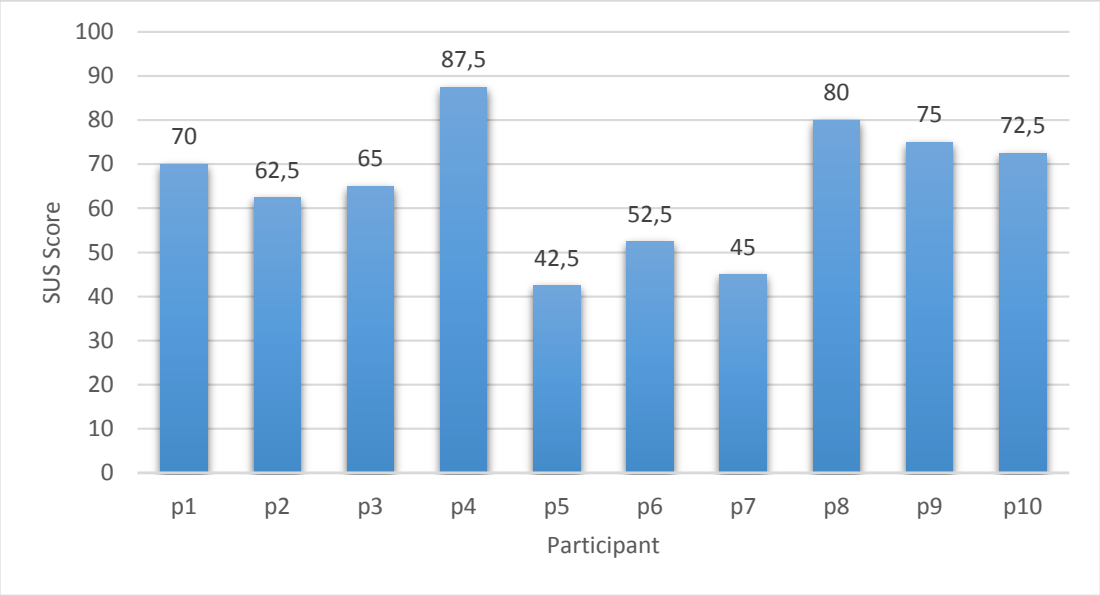


Figure 5.4: Individual SUS scores for Mouse Interaction

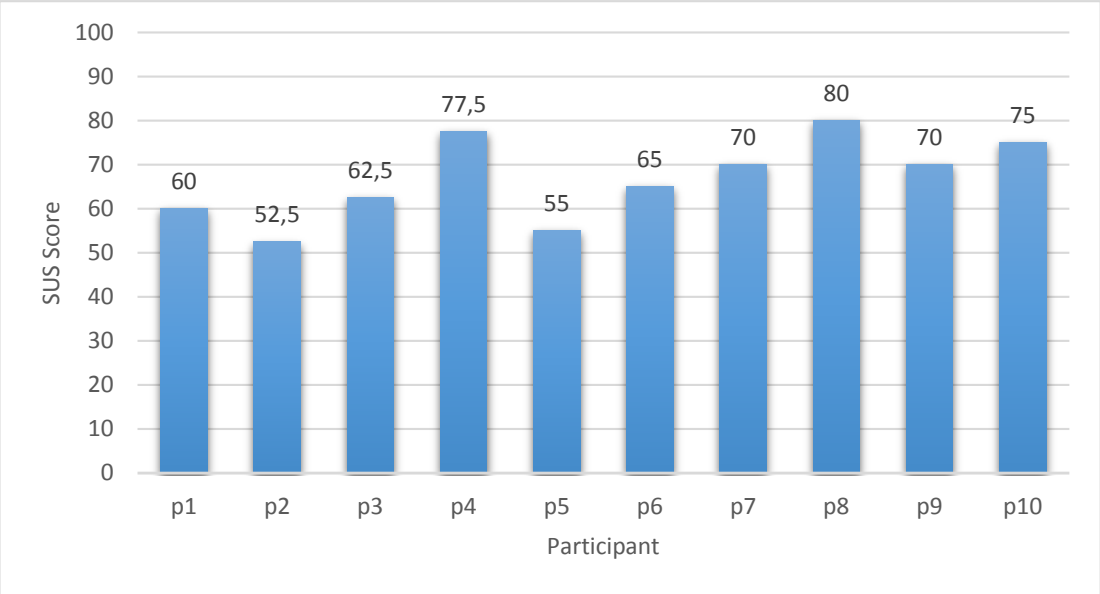


Figure 5.5: Individual SUS scores for Kinect Interaction

Interface	**SUS Score Arithmetic Mean**	**SUS Score Geometric Mean**	Variance
Mouse	65.3	63.6	198.1
Kinect	66.8	66.1	78.8

Table 5.2: SUS Arithmetic Mean, Geometric Mean and Variance

the Kinect implementation was changed to account for the feedback received from the previous test. Clicking precision was improved by adding input filters and the ability to click by using the left hand while navigating with the right. Seeing how the Kinect's score differs so much between the tests, it appears that the Kinect's reception is very sensitive to the factors just mentioned.

Evaluating the Geometric Mean

The geometric mean is included to prevent outliers from having too much of an impact. Also shown in table 5.2, the geometric means differ from the arithmetic means by only a few points. This could indicate a certain amount of consensus amongst the participants, suggesting that the interface is consistent. Again, the Kinect scores a little higher on average, 4% to be exact. This does not mean that the Kinect is better as an interaction device, but the lack of deviation shows that neither interface significantly outperforms the other. This will be analyzed further when test execution times are presented below.

Evaluating the Variance

Shown in table 5.2 with the rest of the statistical numbers, the variance was calculated to analyze whether or not participants gave similar scores. As can be seen, variance is a lot higher for the mouse than for the Kinect. This makes sense when taken into account that users have long experience with mouse interaction. They will have developed preferences. Also the Petrel interface is indirectly evaluated, as using the mouse is second-nature to most people. During test execution it is therefore hard to distinguish between evaluating the software and the interaction device. With the Kinect it is more obvious what the participant should provide feedback on, since it is not as ingrained as mouse interaction.

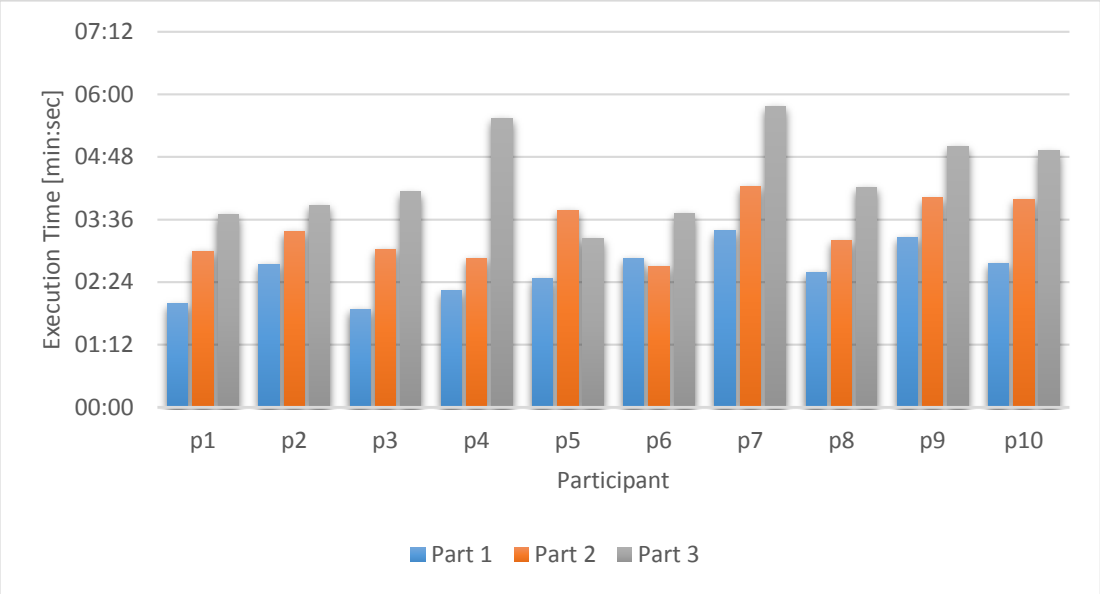


Figure 5.6: Test Execution Time for Mouse Interaction

5.2.2 Time Usage

Figure 5.6 shows execution times with mouse interaction and figure 5.7 shows execution times with Kinect interaction for test parts one, two and three respectively. Again, there does not appear to be a huge difference between the mouse and the Kinect results. This could be attributed to the fact that 90% of the users were unfamiliar with Petrel. None had seen the test tasks before either. The first test featuring mouse interaction therefore went slower, as participants had to figure out the Petrel UI. When using the Kinect, they had already completed the tasks once and knew how the test went and where to find required functionality in Petrel. Still, the fact that the Kinect keeps up is a huge improvement from the testing in Christopher Benjamin Westlye (2014), which featured many of the same operations. Figure 5.8 shows average execution time. When looking at individual test parts, part two stands out. While average time usage for part one and three only differ by nine and two seconds respectively on average, part two average time usage with the Kinect is one minute and 24 seconds longer than with the mouse. Why is this the only part that significantly deviates? Keeping in mind that the low time use is sensitive to errors, steps 10 and 11 in test task two could be the reason. As shown in appendix B, participants had to drag green handles on the 3D model to create a cropped volume. The gesture for doing this was imprecise, so some participants struggled with this when using Kinect interaction.

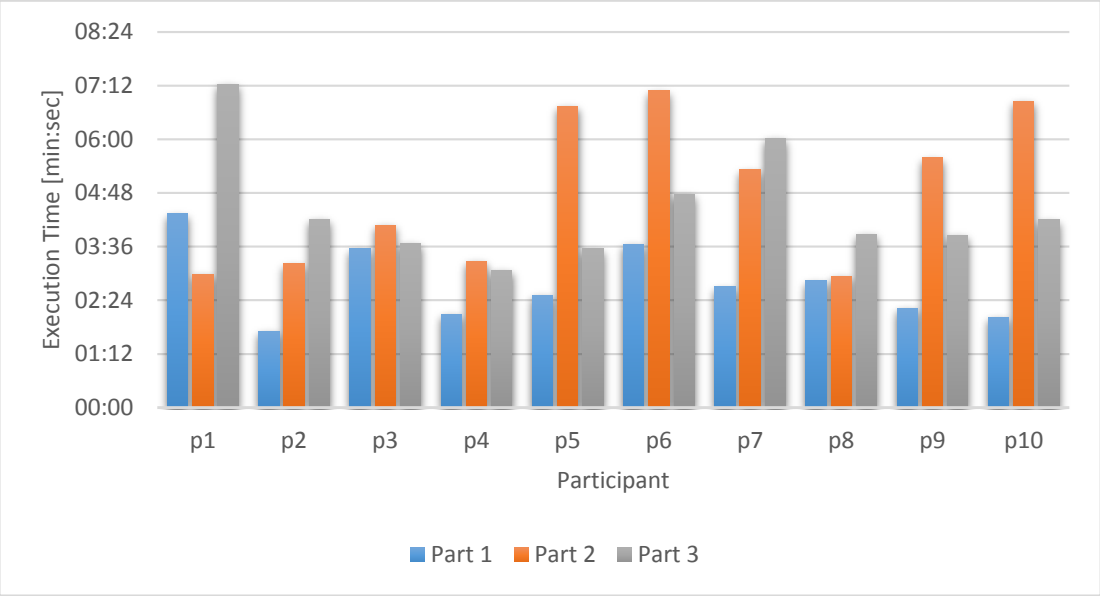


Figure 5.7: Test Execution Time for Kinect Interaction

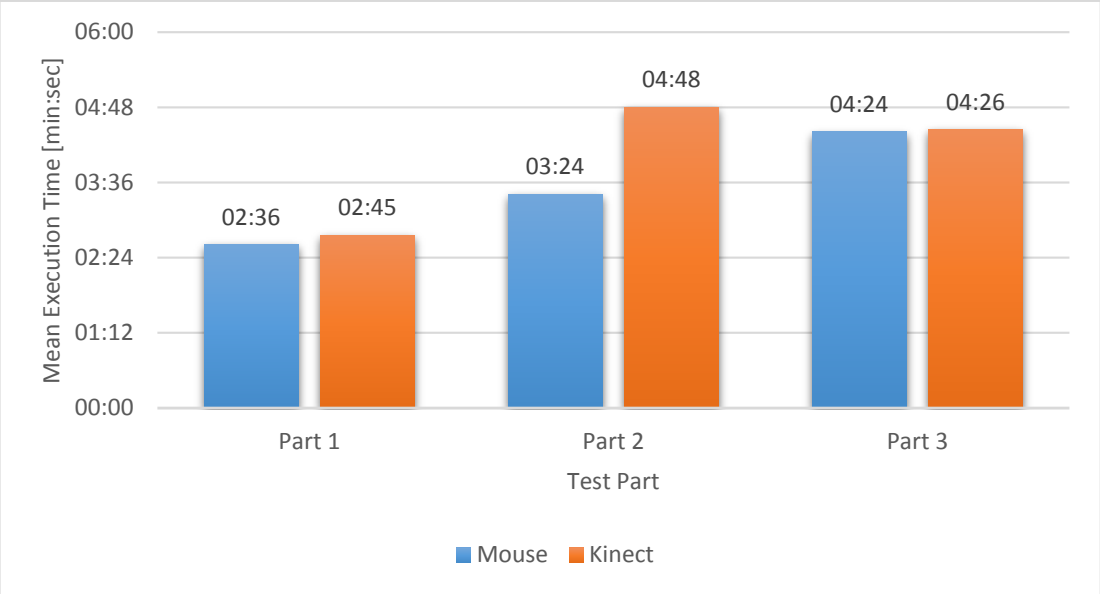


Figure 5.8: Average Execution Time

5.3 Qualitative Results

In this section we present and discuss the participants' post-testing views on effectiveness, efficiency, satisfaction and where the Kinect would be a suitable interaction device in a qualitative way. Participants had to fill out questionnaires both between and after test parts. These in addition to vocal comments gained through the think-aloud method combined with voice recordings are the data used in this section. All questionnaires can be found in appendix A.

5.3.1 Questionnaire Feedback

Which interface was the best with regards to effectiveness?

Asking participants which interface was the best with regards to effectiveness yielded the results shown in figure 5.9. Not surprisingly, the mouse came out on top. One of the responses that were given a lot was that Petrel has a UI focused around mouse interaction. The Kinect could potentially perform more effectively with a UI designed specifically for it. This factor could also have been improved by better precision for not only clicking, but holding. The holding with the right hand was meant for model interaction. When users in test part two had to click and hold a small green point on the seismic, they struggled quite a bit, as mentioned in 5.2.2. Four of the participants used a long time to decide whether to select the mouse or the Kinect as the most effective, reasoning that the mouse only wins due to the user having more experience with it. When combined, more Kinect experience and better click and hold precision could have made the Kinect more effective.

Which interface was the best with regards to efficiency?

Efficiency is different from effectiveness in that it is more about how smoothly and effortlessly the interactions went. Even more users were unsure about what to pick for this category. While everyone thought that the mouse came out slightly ahead in terms of effectiveness, two users actually felt that the Kinect was more efficient. Almost all participants commented that handling the 3D model felt better with the Kinect. Especially rotation was mentioned as a positive experience. Menu and operations requiring precision clicks on the other hand were mentioned as negative experiences with the Kinect. Despite having implemented precision clicking and

filtering, holding one's hands in front of the body as shown by the figures in section 4.3 was reported by users to be tiring. After longer periods of time, this tiredness resulted in decreased precision and increased irritability upon making errors.

Which interface was the best with regards to satisfaction?

All participants rated the Kinect interaction as the most satisfying, also shown in figure 5.9. The most commonly given reason was that it was a new and exciting way to interact with one's work, even for the participants who had tried it before. Several users reported feeling more in contact with the seismic model when using the Kinect. Combined with virtual reality technology like the Oculus Rift, this could be a step closer to making abstract data feel more intuitive. Three users also mentioned that completing a task with the Kinect felt like an accomplishment, stimulating the users to interact more as if playing a game. It is possible that this stimulation would decrease with more exposure to the Kinect. Nonetheless, the fun factor of the interaction should not be disregarded. Keeping the user interested in the data could be beneficial in regards to work productivity. So despite reporting the Kinect as tiring and lacking in precision when compared to the mouse, there was a general consensus that it is an interaction device that provides a satisfying experience. Again, it can be speculated that the levels of enthusiasm shown by testers would be reduced had they used Kinect as much as they have used the mouse.

Where would the Kinect be the most suitable option?

Some of the user suggestions for where the Kinect would be a suitable interaction device are shown in the list below. It was emphasized that the area of use should be related to the test that had been performed. Some entries, particularly the one about 3D model manipulation was mentioned by many. A trend in the replies seem to be that the Kinect can be used for interactions requiring large hand movements. For example, it does not matter where in the 3D window the cursor is when rotating, as long as it is inside the window. When it comes to presentations and demonstrations, these are interactions that can be brief. One does not need to use the interaction device for an extended amount of time. This seems to reflect upon the fact that participants reported feeling tired after prolonged Kinect use. In short, users speculated that the Kinect could be suitable for interactions that required low precision and could include

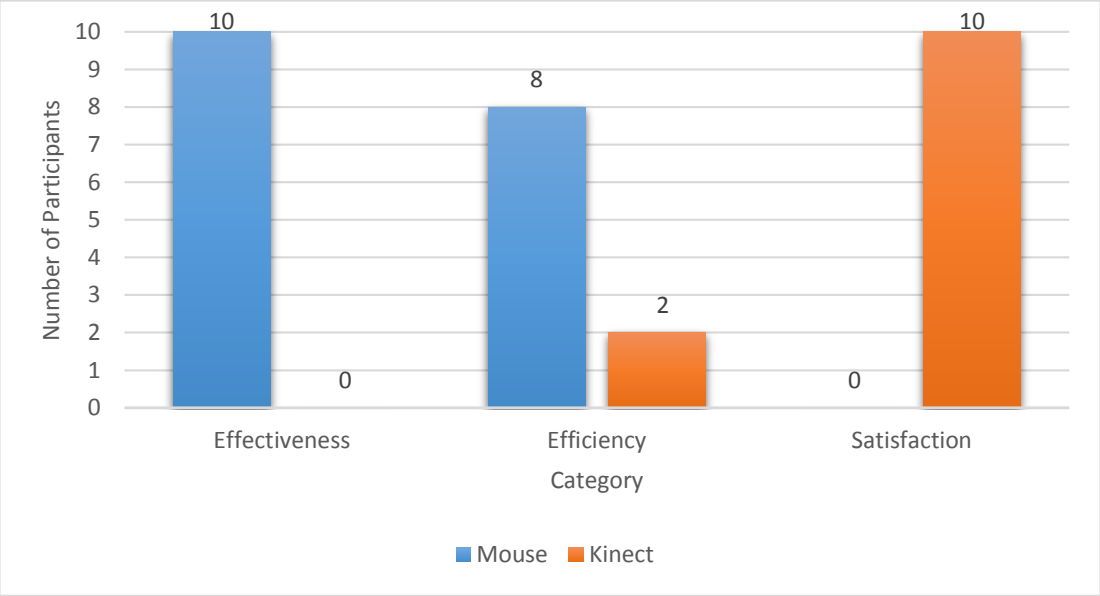


Figure 5.9: Preferred Interface based on Effectiveness, Efficiency or Satisfaction

longer breaks between uses.

- 3D navigation and model manipulation
- Handling 3D models before 3D printing
- Computer interaction as shown in Iron Man movies
- Education: could motivate kids
- Presentations without requiring a physical device
- Demonstration of a product, for example at a stand

5.3.2 Issues and Suggested Improvements

Good precision was reported when using the Kinect for left clicking. Holding the left mouse button down however, was reported as not providing the same levels of precision. This is largely due to how the gestures are implemented, as was discussed in 4.3. This is definitely useful feedback, as it shows that the design decision of making left hold intuitive gesture-wise did not provide the precision levels expected by the users. Another issue that was commented on a lot was the

difficulty of double clicking with Kinect. Having to close one's hand twice in a row at similar speeds to double left clicking a mouse proved challenging to most participants. Some had an easier time than others, but this does not justify the lack of proper double clicking support, so that all users can perform actions efficiently. Other than the issues mentioned, participants did not provide much feedback on how to improve the implementation. Two participants wanted the ability to adjust Kinect sensitivity. This is easily implemented, as the right hand's position is multiplied with a constant to determine the area in front of the Kinect which corresponds to the screen. By providing a gesture for tuning the constant, users can effectively increase the area for precision work. This would then require a larger hand movement to move the cursor, preventing the cursor from moving away from the desired screen position due to some small unintended hand movement.

Chapter 6

Summary and Recommendations for Further Work

This final chapter summarizes the work performed and the results obtained in this project. Section 6.1 attempts to draw conclusion based on the performed analysis. Lastly, section 6.2 makes suggestions for future work.

6.1 Summary and Conclusions

In this master thesis different interaction devices for Petrel workflow execution were compared. Potential usability of the Kinect sensor relative to the established computer mouse interaction device was investigated. Kinect support for Petrel was implemented and is detailed in chapter 4. Test tasks simulating realistic Petrel workflows were created based on material from a Petrel introduction course, as mentioned in 3.2. The comparison was then conducted by performing a usability study. Earlier work (Christopher Benjamin Westlye (2014)) provided a reference for both test implementation and execution. SUS questionnaires, voice recordings, test time tracking and different qualitative methods were employed to properly measure usability. As detailed in section 5.1, ten participants in the age range 21-29 were involved in the study. They were all students in technical or scientific fields. 70% had used the Kinect before while only 10% had used Petrel before. Arithmetic means, geometric means and variance for the SUS test scores were calculated and evaluated in section 5.2. The arithmetic mean was 1.7 points higher

than the geometric mean for mouse scores and 0.7 points lower for the Kinect scores. We can conclude that this lack of variation indicates a consistent interaction experience for the participants. The arithmetic mean for mouse scores was 2.2% higher than for Kinect scores, suggesting that neither interaction device outshines the other in terms of usability when the numbers are looked at in isolation. Variance was 40% higher for mouse SUS scores. People were more used to mouse interaction and their established preferences became apparent through the variance differences as discussed in section 5.2. Test execution times with either device only differed by a few seconds for parts one and three. Test task two is the only exception with its one minute and 24 seconds longer average execution time with Kinect. It can be concluded with relative certainty that this is due to poor precision when emulating the holding of the left mouse button, resulting in difficulties related to steps ten and eleven. Qualitative feedback was provided by study participants through questionnaires and the think-aloud method. In spite of comparable execution times, ten out of ten users rated the mouse as more effective at solving the test tasks. This makes sense when seen together with the numerous participant reports about tired arms after prolonged Kinect use. Somewhat contradictory, two participants reported the Kinect as being more efficient. This is possibly due to the increased freedom of movement the Kinect provides the user. Despite reporting the mouse as more effective and efficient, all users reported the Kinect interaction as more satisfying, claiming that it provided feelings of accomplishment and stimulation. It can be assumed with good probability that this is due to the novelty effect. Especially when looking at user comments about how the Kinect was a fresh and different interaction experience, even if they had used it before. With more use however, it is possible that this effect would be be less pronounced. As it stands, all participants were enthusiastic when sharing their feelings about Kinect.

6.1.1 Final Conclusion

As reflected by SUS scores and execution times presented in 5.3, both interaction devices enabled the users to successfully complete the test tasks. It can be concluded that the Kinect's score is highly dependent upon which gestures are available and how they hold up in regards to the presented tasks. As the Kinect excelled in 3D model interaction and was reported to fall short when navigating menus and options, it is safe to conclude that with a proper implemen-

tation the Kinect is a proper interaction device in some areas. The most commonly mentioned areas were demonstrations and presentations, where the novelty and lack of physical interfaces could make interactions more effective, efficient and most of all, satisfying.

6.2 Recommendations for Further Work

6.3 Short-term

As test part three about data visualization revealed, Kinect needs a proper gesture for double clicking. It is apparent that double clicking the left mouse button can not be directly emulated with one's hand. Clicking the left mouse button quickly mainly uses the muscles in the index finger while closing one's hand to left click with the Kinect requires not only muscles in all the fingers, but in the core of the hand as well. A slightly different problem revealed itself in test part two about data optimization, when the Kinect was to be used for left click hold actions demanding precision. While precision was supported for left clicking, it was not for left holding. This needs to be improved upon. For example by letting the left hand not only left click, but also hold. Coming up with proper gestures takes time, but the implementation itself does not require long-term working. The requested ability to adjust sensitivity should also be implemented, further improving precision when needed and efficiency when not needed.

6.4 Long-term

Using ten participants was enough for this study, but quantitative results could be vastly improved with more test completions. The gesture planning and implementation used a significant amount of time in this project. Working more on defining good gestures that increase precision and prevent users from tiring quickly could be time well spent if one wishes the Kinect to be a competitive interaction device.

Bibliography

Kinect API Overview. http://msdn.microsoft.com/en-us/library/dn782033.aspx. Last visited: 2015-04-30.

Schlumberger Petrel Software. http://www.software.slb.com/products/platform/Pages/petrel.aspx. Last visited: 2015-03-02.

XAML Overview. http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx. Last visited: 2015-02-25.

Alabbasi, N. (2013). Usability Study of the Petrel Software: Comparing the Three Interfaces Based on WIMP, Ribbons and Touch. Project Thesis, University of Stavanger.

Albrektsen, S. M. (2011). Using the Kinect Sensor for Social Robotics. Master's Thesis, Norwegian University of Science and Technology.

Aqrawi, A. A. (2013). Accelerating Disk Access Using Compression for Large Seismic Datasets on Modern GPU and CPU. Master's Thesis, Norwegian University of Science and Technology.

Barnum, C. M. (2010). *Usability Testing Essentials: Ready, Set... Test!* Elsevier Science & Technology.

Brooke, J. (1996). SUS: A quick and dirty usability scale.

Caroll, J. M. (2004). Human Computer Interaction - Brief Intro. In Soegaard, M. and Dam, R. F., editors, *The Encyclopedia of Human-Computer Interaction*. The Interaction Design Foundation, Aarhus, Denmark, 2nd edition.

Christopher Benjamin Westlye (2014). A Usability Study Featuring Microsoft Kinect for Windows v2. Project Thesis, Norwegian University of Science and Technology.

Cohen, M. B. (2004). *Designing Test Suites for Software Interaction Testing*. PhD thesis, The University of Auckland.

Faulkner, L. (2003). Beyond the Five-User Assumption: Benefits of Increased Sample Sizes in Usability Testing. *Behavior Research Methods*, 35:379–383.

Hsi, I. and Potts, C. (2000). Studying the Evolution and Enhancement of Software Features. In *Proceedings of the 2000 IEEE International Conference on Software Maintenance*, pages 143–151.

International Organization For Standardization (1998). *ISO 9241-11: 1998: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) - Part 11: Guidance on Usability*. International Organization for Standardization.

John Lester Sarmiento Gerardo (2007). The Effectiveness of Novice Users in Usability Testing. Master's Thesis, University of Oslo.

Juliana Anyango Ogolla (2011). Usability Evaluation: Tasks Susceptible to Concurrent Think-Aloud Protocol. Master's Thesis, Linköping University.

Karray, F., Alemzadeh, M., Saleh, J. A., and Arab, M. N. (2008). Human-Computer Interaction: Overview on State of the Art. *International Journal on Smart Sensing and Intelligent System*, 1:137–159.

Kean, S., Hall, J., and Perry, P. (2011). *Meet the Kinect: An Introduction to Programming Natural User Interfaces*. Apress.

Microsoft. Xbox One Official Site. http://www.xbox.com/en-US/xbox-one. Last visited: 2015-01-27.

Microsoft (2014). Human Interface Guidelines v2.0. http://go.microsoft.com/fwlink/?LinkId=403900.

Moore, G. E. (1965). Cramming More Components onto Integrated Circuits. In *Electronics*, pages 114–117.

Myers, B. A. (1998). A brief history of human computer interaction technology. *ACM interactions*, 5:44–54.

Nielsen, J. (1993). *Usability Engineering*. AP Professional.

Nielsen, J. (2012). Usability 101: Introduction to Usability. http://www.nngroup.com/articles/usability-101-introduction-to-usability/. Last visited: 2015-05-21.

Pavlovic, V. I., Sharma, R., and Huang, T. S. (1997). Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:677–695.

Records, G. W. Fastest-selling gaming peripheral. *Guinness World Records*.

Rubin, J. and Chisnell, D. (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. Wiley, Indianapolis, Indiana.

Schlumberger (2014). *2014 Petrel Geophysics - Seismic Visualization and Interpretation: Training and Exercise Guide*. Schlumberger.

van Someren, M. W., Barnard, Y. F., and Sandberg, J. A. (1994). *The Think Aloud Method: A Practical Guide to Modelling Cognitive Processes*. Academic Press.

Virzi, R. A. (1992). Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough? *Human Factors*, 34:457–468.

Wirth, N. (1995). A Plea for Lean Software. *Computer*, 28(2):64–68.

Appendix A

Usability Test Questionnaires

Data Collection Before Test

Demographic Information

Name:

Gender:

Age:

Educational Background:

Current Position:

Experience

Do you use desktop/laptop computers on a daily basis (with keyboard and mouse)? If not, specify usage.

Do you work with any kind of task based software in your job / studies? Please elaborate.

Have you used Petrel before? If yes, please specify usage.

Have you used the Kinect sensor or similar interfaces before? Please elaborate.

Data Collection During Mouse Test

Task Performance: Success/ Fail / Partial success.

Total Completion Time (min):

Task 1 Completion Time (min):

Task 2 Completion Time (min):

Task 3 Completion Time (min):

Perceived User Satisfaction (Moderator point of view):

- Fun or boring
- Annoying or comfortable
- Complicated or simple
- Faster or slower

Data Collection During Kinect Test

Task Performance: Success/ Fail / Partial success.

Total Completion Time (min):

Task 1 Completion Time (min):

Task 2 Completion Time (min):

Task 3 Completion Time (min):

Perceived User Satisfaction (Moderator point of view):

- Fun or boring
- Annoying or comfortable
- Complicated or simple
- Faster or slower

Data Collection After Mouse Test

		Strongly disagree 1	2	3	4	Strongly agree 5	Supporting comments
1	I think that I would like to use this interface frequently						
2	I found the interface unnecessarily complex						
3	I thought the interface was easy to use						
4	I think that I need the support of a technical person to be able to use this interface						
5	I found the various functions in the interface well integrated						
6	I thought there was too much inconsistency in the interface						
7	I would imagine that most people would learn to use this interface very quickly						
8	I found the interface very cumbersome to use						
9	I felt very confident using the interface						
10	I needed to learn a lot of things before I could get going with this interface						

Data Collection After Kinect Test

		Strongly disagree 1	2	3	4	Strongly agree 5	Supporting comments
1	I think that I would like to use this interface frequently						
2	I found the interface unnecessarily complex						
3	I thought the interface was easy to use						
4	I think that I need the support of a technical person to be able to use this interface						
5	I found the various functions in the interface well integrated						
6	I thought there was too much inconsistency in the interface						
7	I would imagine that most people would learn to use this interface very quickly						
8	I found the interface very cumbersome to use						
9	I felt very confident using the interface						
10	I needed to learn a lot of things before I could get going with this interface						

Data Collection After All Tests

Which interface was the best with regards to effectiveness (Being able to complete given tasks)? Why?

Which interface was the best with regards to efficiency (Being able to complete tasks quickly/easily)? Why?

Which interface was the best with regards to satisfaction (Enjoying the interaction)? Why?

Can you think of a situation where the Kinect would be the most suitable option? Please explain

Other comments:

Appendix B

Usability Test Tasks

Task 1: Gain Familiarity

1. Open a 3D window from the **Insert** group on the **Home** tab
2. In the **Input** pane, find the Survey subfolder in the Seismic mainfolder
3. Toggle on **SeismicTestData**, **Inline**, **Xline** and **Z** (in that order)
4. Click the **View** option on the window toolbar
5. Use zooming and scrolling to navigate around the 3D object
6. Zoom in enough so that the object fills the 3D window
7. Rotate the object so that the directional arrow points horizontally with the green side up
8. Close the 3D window

Task 2: Data Optimization

1. Open a 3D window from the **Insert** group on the **Home** tab
2. In the **Input** pane, find the Survey subfolder in the Seismic mainfolder
3. Click the seismic data **SeismicTestData** in the Survey folder, making the **Tools** tab appear

4. Open the **Tools** tab
5. In the **Operations** group on the **Tools** tab, click **Prefetch to cache**
6. Toggle on **SeismicTestData**, **Inline**, **Xline** and **Z** (in that order)
7. In the **Operations** group on the **Tools** tab, click **Insert virtual cropped volume** with **SeismicTestData** selected
8. A new icon with the same name as the original seismic data followed by [Crop] 1 appears below the original icon
9. Activate **Select** mode from the window toolbar
10. Click and drag the green handles to move the edges of the cube manually
11. Make the cropped volume approximately 1/4 the size of the original volume
12. Make sure **SeismicTestData [Crop] 1** is selected
13. In the **Operations** group, click **Realize seismic**
14. Click **Realize** on the bottom of the popup window
15. Click **Apply** and **OK**
16. Close the 3D window

Task 3: Data Visualization

1. Open a 3D window from the **Insert** group on the **Home** tab
2. With the 3D window open, toggle on **SeismicTestData [Crop] 1 [Realized] 1** (created in task 2), **Inline** and **Xline** and **Z** (in that order)
3. Click the **View** option on the window toolbar
4. Rotate the object so that the directional arrow points horizontally with the green side up

5. Click **Inspector** from the **View** group on the **Home** tab
6. Click the **Select** option on the window toolbar
7. Click the 3d object to activate the inspector tool for the object
8. View and expand the **Colors** tab that appears in the **Inspector** box
9. under **Colors**, change the color table to **Seismic dip azimuth**
10. Click the **View** option on the window toolbar
11. Rotate the object so that the large green area is clearly visible
12. With **SeismicTestData [Crop] 1 [Realized] 1** selected, Open the **Tools** tab and choose **Arbitrary polyline** in the **Create intersection** group
13. Create four points on the green area
14. Double click somewhere on the green area
15. Click the **View** option on the window toolbar
16. Zoom in on the newly created polyline so that the whole 3D window turns green
17. Close the 3D window