Usability of natural user interface buttons using Kinect

Tommi Pirttiniemi

University of Tampere
School of Information Sciences
M.Sc. Thesis
Supervisor: Roope Raisamo
December 2012

University of Tampere
School of Information Sciences
Interactive Technology
Tommi Pirttiniemi: Usability of natural user interface buttons using Kinect
M.Sc. Thesis, 45 + 7 pages
December 2012

Abstract

The Kinect device opened a world of new possibilities for developers and user interface designers to explore. With Kinect, users can interact with user interfaces by using just their hands and body. A typical example of a natural user interface, or NUI, is an Xbox 360 game that uses Kinect as its only input method. The PC version is faster and more accurate than its Xbox 360 counterpart. This encouraged us to research ways and methods how we could utilize it in terms of improved usability. Unfortunately, the NUI takes us a step backwards in terms of usability, since no haptic feedback is available when interacting with it. The theory of NUI is a decades-old research field, but with the advances in technology in recent years, it has finally become a reality for the consumers.

Using just your body to interact with the NUI is slow and sometimes error-prone compared to the classic mouse and keyboard interaction. Our main research question was how to speed up the interaction with NUIs and still keep it easy, accurate and nearly error-free.

We chose two of the most commonly used NUI interaction types as our comparison points, and we developed nine new interaction types as our proposals. We ran usability tests of 20 participants and recorded the completion times for each of the interaction types and also the error rates.

Based on the usability test results, the two-handed push button was the fastest of the interaction types, and at the same time almost error-free. The majority of the participants also chose the two-handed push button as their favorite interaction type.

Keywords: Kinect, natural user interface, usability study, hand gestures, image algorithms, human-computer interaction

Contents

1. Introduction

The Kinect device add-on for Xbox 360 gaming console was released in November 2010, and in a way, it revolutionized how users can interact with user interfaces (UI). Until now, users were tied to at least some kind of hardware device to interact with UIs. Nintendo Wii took a step in the direction of freeing users from hardware's shackles, but it still needed a Wii Remote or more commonly, a Wiimote, for user interaction. PlayStation Eye added controller-free interaction to consoles but only a few games supported it, since the interaction was based only on normal camera and image recognition algorithms.

The Kinect device took the world by surprise with its features, and with a price tag of 200 euros or less, almost every gaming console oriented consumer could afford it. Using Kinect, users can interact with user interfaces by using just their hands, body and voice. This is called *natural user interface* (NUI). Now, with the increasing popularity of the Kinect device, the need for an improved NUI is greater than ever.

In this thesis, we focus on developing and designing ways on how to interact with NUIs and at the challenges they contain [Norman and Nielsen, 2010]. UIs designed especially for Kinect devices are usually based on a special button type that we call a *hover button*. The hover button is activated simply by holding the cursor over the button for a few seconds. Few games also use a second, already existing special button type that we call a confirmation hover button [Nielsen, 2010]. Activating the confirmation hover button requires one additional step: holding the cursor over the confirmation hover button reveals a secondary button, which works like the hover button. Finally, some Kinect-based games, such as Dance Central [Harmonix, 2010], feature buttonless or swipe-based UI.

Almost all of the existing Kinect-based NUIs focus on body and arm gestures rather than more fine-tuned gestures, such as hand and finger movements. Our focus in this thesis is, however, on hand gestures. We used hand position coordinates to control the cursor in the Windows environment, so arm and body movement gestures were left out of our scope of our study.

A major research challenge was to keep the number of false and missed activations to their minimum. In *false activation*, the user does not try to interact with one of our interaction types, but the user's natural movements are misinterpreted as an activation of the interaction type. In *missed activation*, the user tries to interact with one of our interaction types but the system does not recognize it [Dix, 2002].

We proposed several new button types to improve the usability, accuracy and speed of user interaction in Kinect-based NUIs. Programming and user testing were

done in cooperation with Matti Ollila, later referred as M. O. His thesis focuses on the technological aspects, challenges and solutions in the NUI programming. The main focus of my thesis lies in usability of the buttons on the Kinect-based NUIs rather than the technical aspects behind them. Our usability test program contained nine new proposed button interaction types. Two of them were left out from the final user tests based on the feedback we received from the pilot testing. The final seven interaction types were evaluated in a 20-participant usability test.

We hypothesized that our new interaction type, which we call the *two-handed push button*, will perform better than the rest of our tested interaction types. Our second hypothesis is that another of our new interaction types, *the confirmation button*, will be faster than the *confirmation hover button*, and at the same time the error rate will be at acceptable level. Our third hypothesis is that the *L-gesture*-based interaction types will be a viable alternative to our main new interaction type, the two-handed push button.

In the second Chapter, we will first briefly explain what a NUI is and how the existing Kinect-based user controls work. In the Section 2.3, we will go through the different control gestures, including body, arm and hand gestures. Chapter 3 describes our research in more detail. Section 3.1 is about introducing our proposed new interaction types and how they were selected and Section 3.2 is a recount on the usability testing, its setup and the procedure. We take a high-level look into what Kinect-based natural user interface programming is all about. In the Chapter 4, we lay out the results from our usability tests. Existing interaction types are compared to our new proposed interaction types. Finally, in the Chapter 5, we will discuss the results, some scenarios how our findings could be utilizes, and topics for future research. Chapter 6 is the conclusion of our research and discussed topics.

2. Natural user interface interaction

The human interface guidelines document [Microsoft, 2012], later referred to as *the HIG document*, is the current standard for the NUIs that use the Kinect device. The HIG document was released by Kinect team after our designing and developing was done, so we weren't able to take their findings into consideration. Only time will tell if these guidelines will form to be an actual standard for NUIs.

However, as Nielsen [2010] states, there are no universal standards for gestural interactions. This statement has positive implications for our research. Even though some of our research conflicts with the dos and don'ts of the HIG document, it is merely a guideline and can be challenged. For example, the HIG document recommends using only symmetrical two-handed gestures, and that the two-handed gestures are only for advanced-user and non-critical tasks. However, we decided to use two-handed gestures for button activations, which can be considered a critical task.

2.1. Natural user interface

The natural user interface, or NUI, is the third step in the evolution of the user interfaces. The first commonly used user interface was the command-line interface (CLI) which was used for human-computer interaction (HCI) before the graphical user interfaces. Only the keyboard is used to interact with the CLI and the actual UI consists entirely of symbols. The CLI is still used in some operating systems, such as UNIX, but graphical interfaces have superseded it decades ago. The second step of the user interface evolution was the graphical user interface (GUI). GUIs are nowadays the most common UI type, which allows user interaction using images rather than text commands. Any major personal computer operating system, such as Windows 7, can be considered a typical GUI interface.

User interfaces can also be categorized using less familiar acronyms, such as the reality user interface (RUI) [Mann, 1998], the organic user interface (OUI) [Vertegaal and Poupyrev, 2008] or the kinetic user interface (KUI) [Bruegger and Hirsbrunner, 2009]. The NUI Group Community [2009] describes the term natural user interface as:

> "…an emerging computer interaction methodology which focuses on human abilities such as touch, vision, voice, motion and higher cognitive functions such as expression, perception and recall. A natural user interface or "NUI" seeks to harness the power of a much wider breadth of communication modalities which leverage skills people gain through traditional physical interaction".

The NUI concept is fairly large, and includes parts, such as touch, that are outside the capabilities of the Kinect device. We would have loved to include touch as part of our research, but it was not possible at the current level of technology. The closest to touchable holograms are airborne ultrasound tactile displays [Iwamoto *et al.*, 2008], but the technology is still in its infancy. In our research, we have solely focused on the NUI subcategories of motion and computer vision.

2.1.1. Cursor-based or cursorless interfaces

The modern personal computer UIs are, almost without an exception, cursor-based interfaces, where the cursor is used to interact with the GUI. Console-oriented UIs, on the other hand, are usually cursorless interfaces, where a game controller is used to navigate cursorless menus. Nintendo Wii was the first console to bring the cursor-based UI and navigation back to consoles, and soon Xbox 360 offered an alternative cursor-based dashboard UI for the Kinect device. Although the Kinect-based UI for the Xbox 360 dashboard can be used with both the Xbox controller and the Kinect device, it is mainly designed for the latter.

Cursor-based user interfaces also have sub-categories, such as the traditional PC interface that are used with a mouse. Other cursor-based UIs are mainly extensions of the traditional interfaces, such as the hover button. These extensions can either be on the UI's side, or on the controller device's side. A UI extension can be, for example, a special button that is activated automatically after the cursor has hovered over the button a for specified time. A controller side extension can be, for example, a special hand gesture that is used to activate a normal button.

Freeman [2010] reflects on the solutions Harmonix [2010] made in their NUI and states rules for making better gestural UIs for Kinect. Like the HIG document [Microsoft, 2012], these rules also contradicts with our research and results. Freeman [2010] quite boldly lays out rules such as "don't use a cursor" and "arm extension does not work". We felt that these rules were quite absurd and we were delighted to prove them wrong.

Almost every cursorless UI is different from one another, some being easier, some harder to use. The cursorless UI offers a great challenge for the developers and designers, since there are no well-established standards. Every user control on the cursorless UI must be developed separately, which greatly increases time required for the development. One of the most successful cursorless NUIs is the menu navigation in the Xbox 360 game Dance Central by Harmonix [2010] and thoroughly covered by Nutt [2010] and Nitsch [2011]. It features a swipe-based natural user interface, which is covered in the subchapter 2.2.3. We chose the cursor-based UI as our research target, since we felt that it had more potential real world applications in the Windows-based environment. For example, some of our proposed new interaction types could be used to interact with the calculator for Windows with little or no modification.

2.2. Kinect-based user controls

We studied the existing special button types developed specifically for the Kinect-based NUI. There are three major interaction types that are in use at the time of this research: a hover button, a confirm hover button, and a swipe button.

2.2.1. Hover button

Nielsen [2010] describes the functionality of the hover button quite accurately: "hold your hand still over a button, while an animated circle is completed." The hover button is the standard in the Kinect-based NUIs, where the cursor is animated instead of the button. This functionality can be observed, for example, in the Kinect-based version of the Xbox 360 dashboard. Several of the Xbox games that use Kinect also use this interaction type.

The hover button in Kinect-based NUI is not designed for repeated actions. If the cursor leaves the hover button area before the animation is complete, the button activation is cancelled, and the animation is stopped. According to the HIG document, the slow click pace of the hover buttons gets in the way of fluent user interaction, and the user must wait for the animation to finish before they can repeat the action. In addition, the cursor is required to leave the button area before the button can be activated again. This makes the hover button very inefficient on the UIs that require multiple button activations. If the hover button is modified to resume the animation when returning to the hover button area, the possibility of a false activation is too great.

Figure 1. Hover button animation

The hover button functionality can be built into a special cursor, or into a special button. It works in a normal button type environment when using the special cursor, or in a program with the special buttons using the normal cursor. It is also one of the fastest interaction types to learn. Although the possibility of activating the hover button by accident is very low, it can still happen.

Our version of the hover button functionality is the same as the original version. However, we changed the animation to be in line with our other interaction types. The animation was changed from cursor animation to button animation. The fill duration of the animation is approximately the same for the button animation as it is in the actual Kinect-based cursor animation, 1.5 seconds.

2.2.2. Confirm hover button

Nielsen [2010] describes the interaction type we call the confirm hover button as follows: “first select a command, and then keep your hand still over a small confirmation button that pops up next to that command”. Without interaction, the confirm hover button is identical to the hover button. However, the confirm part is revealed after the cursor lands on the button area. This additional step slows down the user interaction compared to the hover button. The confirm hover button was the slowest button interaction type in our existing button interaction type review, but also had the least false button activations.

This button interaction type is only used in a handful of NUIs, such as in the game called Your Shape: Fitness Evolved [Ubisoft, 2010]. The confirm hover button in the previously mentioned game works exactly the same as it does on our version of the confirm hover button.

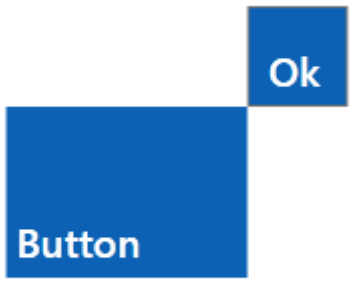


Figure 2. Confirm hover button when the cursor is over the button

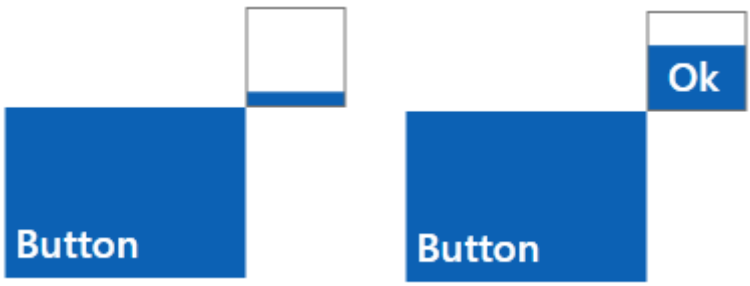


Figure 3. Confirm hover button animation when the cursor is over the Ok button

2.2.3. Swipe button

The third major UI navigation type we call the swipe button is not really a button at all. Swipe button is a special type of cursorless navigation where the menu items are browsed and activated through horizontal and vertical swipe movements. Nielsen [2010] describes the swipe button as: “after selecting a menu item, swipe your hand left — unless you want the "back" feature, in which case you swipe right”.

After prototyping with the special swipe-based user interface, we left the swipe button out of the scope of our research, since cursorless and buttonless interfaces weren't comparable with cursor-based and button-based interfaces.

2.3. Gestures

We have always used gestures for communication and to complement our spoken language. Gestures are a form of non-verbal communication that consists of the movements of the human body and they originate from natural interaction between people according to Kortum [2008]. In this thesis, we will focus on human-computer interaction using gesture interfaces and, more specifically, the Kinect-device.

Gesture-based interfaces are usually thought as the alternative interaction type for classic interfaces but this is not the case with the Kinect device. The user body gestures are the main interaction type in Kinect-based NUIs.

According to the HIG document [2012], gestures can be divided into two categories, the innate and the learned gestures:

> "Innate gestures are ones that the user intuitively knows or that make sense based on the users' understanding of the world. Learned gestures are ones that the user must be taught in order to know how to use them to interact with the system."

Our proposed interaction types contain both of these gestures.

Another way to categorize gestures is to divide them into technology and human-based gestures. Kortum [2008] describes them as:

> "A typical approach to defining an application's gesture vocabulary is to make it easy for the computer system's recognition algorithm to recognize the gestures. The result of this approach for finding gestures can be called a technology-based gesture vocabulary. These gestures together create a vocabulary that might be used where there is no particular meaning to the gesture itself; in brief, the association between gesture and meaning is arbitrary. As an alternative, the human-based gesture approach investigates the people who are going to use the interface, and makes use of human factors derived from HCI research, user-centered design, ergonomics and biomechanics."

Most of the reviewed arm gestures are human-based gestures, and all of the reviewed hand gestures are technology-based gestures.

2.3.1. Body gestures

Although the body gestures are not really suitable for the traditional UIs, they are the bread and butter of interaction in Kinect-based games. The body gestures include all of

the user body stances, including arm positions. The body gesture interaction is also something that the Kinect hardware supports directly. The interaction is made possible by the SDK, which provides the 20 most important joints of the human body that are called skeleton points. These skeleton points include joint positions, such as the shoulder and hand, which can be used in a variety of ways in the NUIs. One UI could track the user's head movements, another could focus just on the hand movements. The body gestures include all the gesture subcategories, such as arm gestures and hand gestures which are covered next.

2.3.2. Arm gestures

Arm gestures are one of the many ways to interact with the NUIs. The arm gestures include all natural user interactions where only the user's arm movements are tracked. The body and hand movements are not part of the arm gestures definition. The user's personal opinion on offensive and inoffensive arm gestures is greatly affected by the cultural differences, but still in lesser amounts than with hand and finger gestures.

Like the body gestures, the arm gestures are also subject to passive gestures. All of the users' natural movements when not trying to interact with a NUI can be considered as passive gestures. Passive arm gestures and arm positions can tell much of the person's current state of mind. For example, the user could express disappointment by placing their hand over their face. The information from the passive arm gestures can potentially be used to build more accurate and advanced NUIs, where all information about the user is taken into account.

Kinect-based games use arm gestures extensively in the gaming environment where cursor is not used or needed. The push gesture is the only gesture suitable for cursor-based user interfaces, but it, too, has its limitations that are discussed in later Chapters.

The Kinect for Xbox 360 introduced a few new arm gestures, such as the stop gesture. The stop gesture is used in a gaming environment to get back to the Xbox dashboard. Although this two-handed arm gesture is easy to use, it is not intuitive. It was designed so that false activations are very hard to make but at the same time it is still very easy to use.

Other commonly used arm gestures in Kinect-based natural user interfaces are the following:

Swipe

In swipe gesture, the hand is moved from one side of the body to the other horizontally. This gesture can also be vertical. Both horizontal and vertical swipe gestures are usually used together in cursorless NUI menu navigation. Swiping from right to left commonly cancels an action or goes to the previous menu screen. Swiping from left to right

commonly selects a menu item or goes to the next menu screen. Swiping up and down commonly browses a list of menu items.

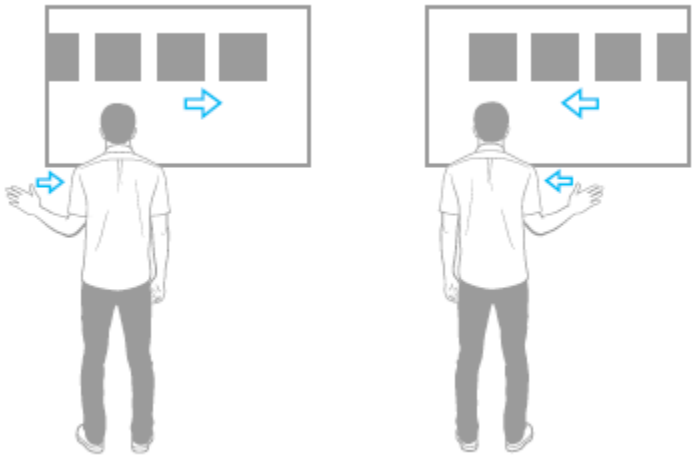


Figure 4. Swipe button [Microsoft, 2012]

Push

In the push gesture, the arm and hand are extended towards the Kinect device. The push gesture belongs to the innate gestures category, and it is the first gesture that users usually try when interacting with Kinect-based NUI without instructions [Holmes, 2011]. The push gesture is different from other arm gestures because it can potentially be used to simulate a mouse click in cursor-based UIs. Other arm gestures are not suitable for cursor-based UIs because the cursor control requires exclusive usage of the hand's vertical and horizontal movements. The push gesture uses only the Z-axis, or depth, and thus it can be used with the cursor. The push gesture is inaccurate and can be frustrating to use if the same hand is also used to control the cursor. Still, the push gesture can provide a great challenge and fun in gaming environment if utilized correctly.

We chose the push gesture to be part of our usability tests, and included several different variations of it in our pilot usability test, which are covered in the Section 3.2.4.

Figure 5. Push gesture along the Z-axis [Microsoft, 2012]

Circle

In the circle gesture, the hand is moved in a clockwise or anti-clockwise circular motion. This gesture differs from the previously discussed gestures by being a continuous gesture without exact given starting or ending point. This makes the circle gesture perfect for some user control tasks, such as browsing through a long list. List items, for example, can be browsed forward by making the clockwise circle gesture, with the speed of the gesture controlling the browsing speed.

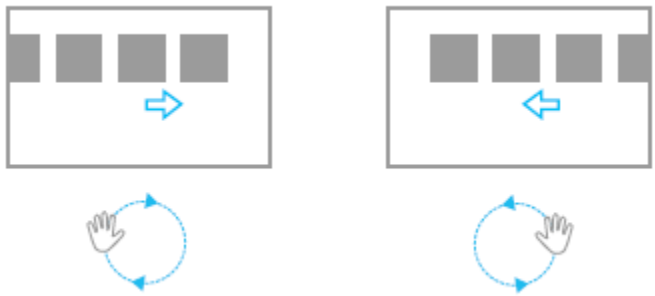


Figure 6. Circle gesture [Microsoft, 2012]

Wave

The wave gesture is the default starting gesture for Kinect-based natural user interfaces. It is used to tell the Kinect which user it should follow. In the wave gesture, the hand is simply waved horizontally a few times. Like the circle gesture, it also is a continuous gesture. Since the wave gesture is the starting gesture, it should not be used elsewhere in NUIs to avoid confusion.

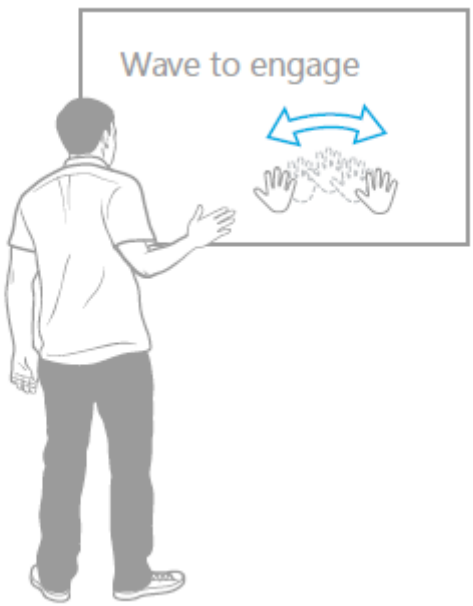


Figure 7. Wave gesture [Microsoft, 2012]

2.3.3. Hand gestures

The hand gestures only include gestures performed without moving your arm or the rest of your body. The push gesture, for example, is an arm gesture rather than a hand gesture, because the push gesture requires the arm to be extended and cannot be done using the hand only. According to the HIG document [2012], like arm gestures, the hand gestures can also be divided into the innate and the learned categories. After an extensive evaluation of different hand gesture types, we decided to include one hand gesture in our usability tests. The included gesture was the L-gesture, which is covered later in this Chapter. Other hand gestures could have been included in the usability tests as well, but we felt that all hand gestures were similar enough so that only one was needed. All of the reviewed hand gestures are covered next.

Closed palm gesture

In the closed palm gesture, the hand is closed into a fist. This gesture is one of the few innate hand gestures. The closed palm was the first hand gesture we evaluated and sought to implement. Unfortunately, due to enormous difficulties in distinguishing the closed palm gesture from the open palm gesture, we had to leave the gesture out our usability tests. One of the key problems we faced was that the hand skeleton point moved when the hand was closed into a fist. It made the hand gesture recognition unreliable and affected the cursor stability.

Figure 8. Closed palm gesture [Microsoft, 2012]

Open palm gesture

In the open palm gesture, the hand is opened while keeping the fingers next to each other without spaces. The open palm gesture is another innate hand gestures. This gesture is quite comfortable to use but also quite hard to identify using any existing algorithms. Our recognition rate was not high enough for this gesture to be included in the usability tests.

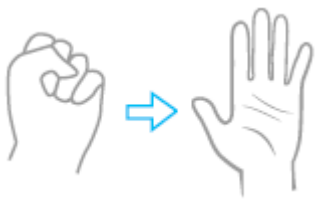


Figure 9. Open palm gesture [Microsoft, 2012]

Five fingers gesture

In the five fingers gesture, the hand is opened while keeping all fingers separated. This gesture belongs to the learned gestures category as well as to the technology-based category, since it has no definite meaning to humans. In general, this gesture is easy for the computer algorithms to recognize and also easy to perform. The five fingers gesture was not chosen to be part of the usability tests, since we felt that there was no connection between a click action and the five fingers gesture.

Figure 10. Five fingers gesture [Microsoft, 2012]

Pointing index finger gesture

In the pointing index finger gesture, only the index finger is extended upwards while keeping other fingers in a fist. This gesture also belongs to the learned gestures category as well as to the technology-based category. We achieved reasonably good success rate for the gesture during our development, but we felt that we could achieve even more accurate recognition by adding extended thumb to the gesture and thus making the L-gesture.

Figure 11. Pointing index finger gesture [Microsoft, 2012]

L-gesture

In the L-gesture, the index finger is extended upwards and at the same time the thumb is extended to the side of the hand while keeping the rest of the fingers in a fist. This gesture should resemble the letter “L”. The L-gesture is yet another gesture belonging to both the learned and technology-based category. This gesture should be easy for our hand recognition algorithm to recognize and at the same time comfortable for all users to perform.

We developed the L-gesture keeping ergonomics in mind. Kortum [2008] covers the range of motion in hand and wrist joints. These limitations were used to keep the L-gesture within natural extension and abduction limits of the hand. Ideally, hand gestures

are not meant to be used for repeated actions. The grander the gestures are, the more they strain the hand. Subtle hand movements can be used more often but our recognition accuracy was not enough for them.

The L-gesture was included in our usability tests for two reasons: to compare it to other new interaction types and to see if a short term use of hand gesture is viable for the click action.

Figure 12. L-gesture

2.4. Summary

In this Chapter, we looked into what NUI really is, and what are the current practices for Kinect-based natural user interaction types. However, the current practices are more like guidelines than well-established standards, and that gave us perfect opportunity to design our own alternatives.

Based on our review, the hover button and the confirm hover button were the most used interaction types, so we chose them as the points of comparison for our new proposed interaction types. We also looked into gestures and how they could be utilized for user interaction. We covered the whole range of human gestures including body, arm and hand gestures, from which we chose one arm gesture and one hand gesture as the basis for our new interaction types.

In the next Chapter, we will cover thoroughly our proposed new interaction types, how we tested them, and how they were developed. We also discuss our research methods and what programming related choices were made.

3. Method

In this Chapter, we describe our work on the Kinect-based NUI type buttons including design, usability tests and programming. New interaction techniques are explained first with an in-depth view of each technique. After that, we cover the usability tests including the two pilot tests, the actual testing, and detailed information on the participants and the actual procedure. Finally, we will take a high-level look into the programming of the Kinect-based natural NUI types. This topic is covered more thoroughly in Matti Ollila's Master's thesis.

3.1. Design

We evaluated many possible interaction types that could be used for interacting with the Kinect-based NUIs. Proposed special button-based new interaction types are covered in Section 3.1.1 and gesture-based new interaction types in Section 3.1.2.

All interaction types were designed to work with either left or right hand used as the main hand, which is used to control the cursor and button activations. This method also works with our two-handed interaction type variants; the main hand is only used to control the cursor, and the other hand only used for button activations.

We divided the proposed new interaction types to three different categories: button types, gesture types and two-handed variants of the button and gesture types.

3.1.1. Proposed button types

The proposed new interaction types are special buttons that can be used with either normal mouse-controlled cursor, or the Kinect device. Gestures can also be used to complement these button types, as can later be seen with *the sticky button*.

Confirm button

The confirm button is a variant of the confirm hover button. We felt that the animation part of the confirmation hover button was redundant and could be left out completely. Instead, we present the confirm button.

The confirm button is visually identical to a normal button at first. When the cursor is over the button area, a smaller button labeled "Ok" appears in the top right corner. When the cursor enters the "Ok" button area, the button is immediately activated. This way, we can control when we want to activate the button and, at the same time, speed up the overall interaction speed.

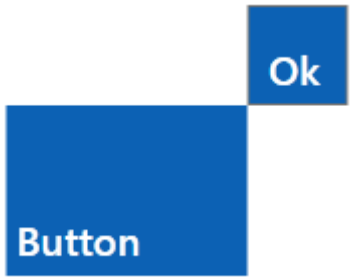


Figure 13. Confirm button

Sticky button

The sticky button is a variant of the snap-to button. A snap-to button is a button type where the cursor is moved automatically over the button when it is near enough. The ordinary snap-to button type was not suitable for our Kinect-based mouse control, so we created a variant of it that we call the sticky button. The sticky button stays focused even if the cursor leaves the button area. This focused button can then be activated anywhere from the screen, not just from the button area. If the cursor enters the area of another button, the focus swaps to that button. All click activations after that are directed to that button.

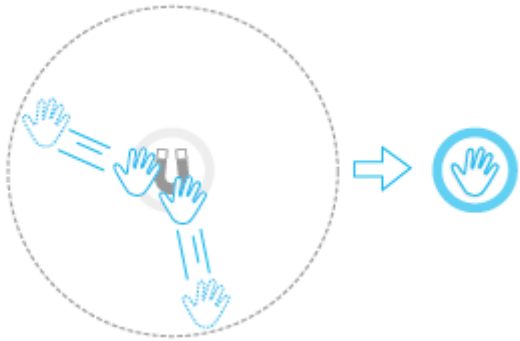


Figure 14. Snap-to button [Microsoft, 2012]

We used the sticky button with the one-handed L-gesture in order to compare the results of the L-gesture versus the sticky L-gesture. The sticky button was also used with the one-handed push gesture in order to compare the results of push button performance versus sticky push button performance.

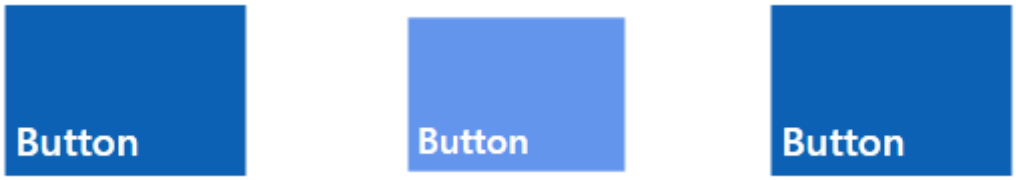


Figure 15. Sticky buttons with the middle one focused

3.1.2. Proposed gesture types

We chose two different gestures for our usability tests: one hand gesture and one arm gesture. Although we did evaluate all of the previously discussed hand and arm gestures, we felt that in terms of usability, the hand gestures in particular were similar enough so that only one was needed for our usability tests.

Based on our internal review during the development, the push gesture itself is not suitable for repeated click actions. The push gesture was still chosen to be part of our usability tests, since we needed a point of comparison for the sticky push gesture and the two-handed push gesture. Another reason for including the push gesture in our tests was to see if introducing the sticky button variant to the push gesture reduces the amount of false activations and improves the interaction speed.

Eventually, we chose the L-gesture as the hand gesture to be used in our usability tests. We decided to include both the normal and the sticky variant of the L-gesture to see if introducing the sticky button to the L-gesture reduces the amount of false activations and improves the interaction speed. The L-gesture was chosen over the pointing index finger gesture because we found it to be more reliable and it had less false activations.

3.1.3. Two-handed variants

The HIG document [Microsoft, 2012] states that one-handed gestures are preferred to the two-handed variants, as they make for a better efficiency and accessibility. However, two of our proposed new interaction types are two-handed. We included these in the tests to see whether two-handed versions are faster and accurate enough to make up for using of both hands.

Two-handed push gesture

In the two-handed push gesture, the main hand is used to control the cursor while the other hand is pushed towards the Kinect device. Technically, the gesture is almost identical to the one-handed version, the difference being that the former is slightly less sensitive. The two-handed push gesture offers us exclusive use of the main hand to control the cursor while the other hand can be idle. The other hand is used only when the user wants to activate buttons. This two-handed variant potentially gives us great accuracy for the cursor control and only a minimal possibility of false and missed activations.

Two-handed L-gesture

The two-handed L-gesture is performed by making the hand gesture with your other hand while your main hand is only used for controlling the cursor. Technically, the L-gesture is the same as in the one-handed L-gesture. Like the two-handed push gesture, the two-handed L-gesture also offers us exclusive use of the main hand to control the

cursor while the other hand can be idle. The user can also focus completely on making comfortable hand gestures with the other hand because the hand movements are not used for controlling the cursor.

3.2. Usability tests

The usability tests were run in the Gaze lab B1071 at the University of Tampere. We showed the premises to the participants after their arrival. First they were asked to sign a consent form, and we did emphasize that they could withdraw their consent and stop the test at any given time. After signing the consent form, we asked them to fill out a questionnaire. Finally, we explained the general test procedure to them step by step [Lewis and Rieman, 1994].

Before each of the interaction type tests, we displayed a training view to the participants. This training view had two clickable buttons and written information about the upcoming interaction type. After one of the training buttons was activated at least once, a button with "Done"-label appeared at the lower right corner. This button started the actual testing and the test recording.

3.2.1. Participants

We chose 20 participants for the actual usability tests as suggested by Nielsen [2006]. We also chose one additional participant for the first, and another one for the second pilot test. We collected relevant background information with a questionnaire that could be used to categorize participants [Sova and Nielsen, 2003]. The first question in our questionnaire was whether the participants had used the Kinect device before. We divided usage into four categories:

often	0
few times	5
once	2
never	13

Table 1. The Kinect device usage and the participant distribution

We excluded minors from the sampling because of the parental consent. As the target audience for Kinect is mainly thirty-year-olds and under, we didn't have any participants over 50. Age groups were then divided into group categories:

18 – 25	3
26 – 35	16
36 – 50	1
over 50	0

Table 2. Age groups and the participant distribution

We also asked participants to rank their computer-related skills into one of the three predetermined categories. High number of participants ranking themselves as experts can be explained by their technical background.

beginner	0
intermediate	11
expert	9

Table 3. Computer experience and the participant distribution

None of the participants reported any kind of disability concerning hand or arm movements, so unfortunately we were not able to study how much disabilities would affect the usability of the Kinect device. The gender of the participants was not asked in the questionnaire, but was still recorded.

male	12
female	8

Table 4. Gender distribution

3.2.2. Apparatus

Our research centers around the Kinect device. We were interested in the Kinect device, and the potential it had for advanced NUIs, as soon as it hit the market. Shortly after the release, there were unofficial hacks that made it possible to attach the Kinect device to a PC and develop Kinect-based applications for Windows and other platforms. The first beta version of the official software development kit, or more commonly, Kinect for Windows SDK, was released in 2011, and we immediately started to experiment with it. Unfortunately, with the release of the official version, the whole application programming interface (API) was redesigned, and we had to rewrite most of the work we had done so far.

As stated in the HIG document [Microsoft, 2012], the technical properties of the Kinect device include an infrared and a RGB camera with 57° horizontal vision and 44° vertical vision, and a practical viewing distance of 1.2 – 3.5 m. Kinect for Windows SDK also contains a near mode, which enables a closer viewing distance. This near mode was included in the SDK after our programming was done, and therefore we didn't include it in our testing. In addition to movement tracking, the Kinect device also contains advanced features of user voice tracking, but we excluded voice as interaction type from our tests.

Figure 16. The Kinect sensor [Microsoft, 2012]

As for the other test equipment, we used a video projector with a resolution of 1024 x 768 to project the screen on the wall. Our test program ran on a Samsung RF711 laptop with Windows 7 operating system, which also functioned as the testing platform for our first part of the test, the mouse training.

We were able to take recommendations of the HIG document [Microsoft, 2012] into consideration when choosing and setting up the test environment for our usability tests. We selected a dedicated windowless room to ensure quiet and private testing, and to prevent sunlight from interfering with the infrared camera on the Kinect device. We took the field of vision of the Kinect device into account when configuring the environment to ensure that the floor and the user were visible to the camera. The test conditions can be considered as the optimal circumstances, and our results could have been notably different in a mixed environment.

3.2.3. Usability test development

When interacting with the Kinect-based NUI, we wanted to give the user all the possible feedback available, since haptic feedback was not possible. That left us two kinds of feedback, auditory and visual feedback. As for the former, typical Windows 7 click sound was played when a button was activated to indicate a successful click. And as for the latter feedback type, our buttons also had a visual indicator when clicked.

According to the HIG document [Microsoft, 2012], test the scenarios are not supposed to focus on productivity, speed or precision. In a way, this contradicts our research question on NUI button interaction types. The Kinect device is almost exclusively used in games that leave their use of the UI and button presses to the bare minimum. This is mainly due to the slow activation speed of the hover button, and the inaccuracy of controlling the cursor with your hand. Our research focuses on advancing from gaming-oriented NUI environment to application environment, where speed and precision of the UI are valued factors.

The interaction with the Kinect device is usually started with the wave gesture. However, we started our user interaction immediately when the user was recognized, since there was no possibility of a wrong user being on the scene in our controlled

environment. In both our setup and the Kinect default setting the NUI is always on after the user has been recognized, and stays on until the user leaves the scene.

Two of the most important user-centered design processes and methods are iterative design and usability evaluation [Vredenburg *et al.,* 2002]. Other methods, we used to gather additional data were user interviews and prototype without user testing. With these four methods, we could obtain statistically significant data.

The iterative design was our primary method when developing the usability test prototype. Each of the interaction types were developed by trial and error until we were satisfied with our work.

After each of our new interaction types was programmed, we used them in our prototype without user testing. Based on this, new interaction types were fine-tuned with our internal prototype, until we achieved a reasonably good success rate with each of the interaction types.

At first, we used the Keystroke Level Model (KLM) [Kieras, 1993] to estimate the time it takes to complete our interaction types. Unfortunately, the KLM applies only to the keyboard and mouse interaction, and at the time of our research, there were no methods available to evaluate Kinect-based NUI interaction speed.

3.2.4. First pilot test

The usability tests were designed with the Modern UI in mind to reflect the upcoming Windows 8 and its theme. The button size was set to 120 x 85 pixels using the screen size of 1024 x 768 as previously mentioned. There were a total of 16 buttons in the test layout.

The first pilot test was mainly designed for the evaluation of which interaction types were suitable for the actual usability tests. There were a total of 11 different new interaction types that we tested, or 12 with the mouse training included. There was a short break after each test to help the participant relax and recover from fatigue. During this break we also explained how the next interaction type worked. The purpose of the mouse training test as the first test was to familiarize the participant to the layout as well as to test the UI before proceeding to the NUI part of the usability test. After the mouse training, the order of the tested interaction types was randomized to eliminate possible learning factors from skewing the results. For example, if the single variant of a gesture button was always tested before its two-handed counterpart, the results would favor the latter.

The tested interaction types in the first pilot testing were:

- Hover button
- Confirm button
- Confirm hover button
- Gesture button

- Push button
- Sticky gesture button
- Sticky push button
- Two-handed gesture button
- Two-handed push button
- Two-handed sticky gesture button
- Two-handed sticky push button

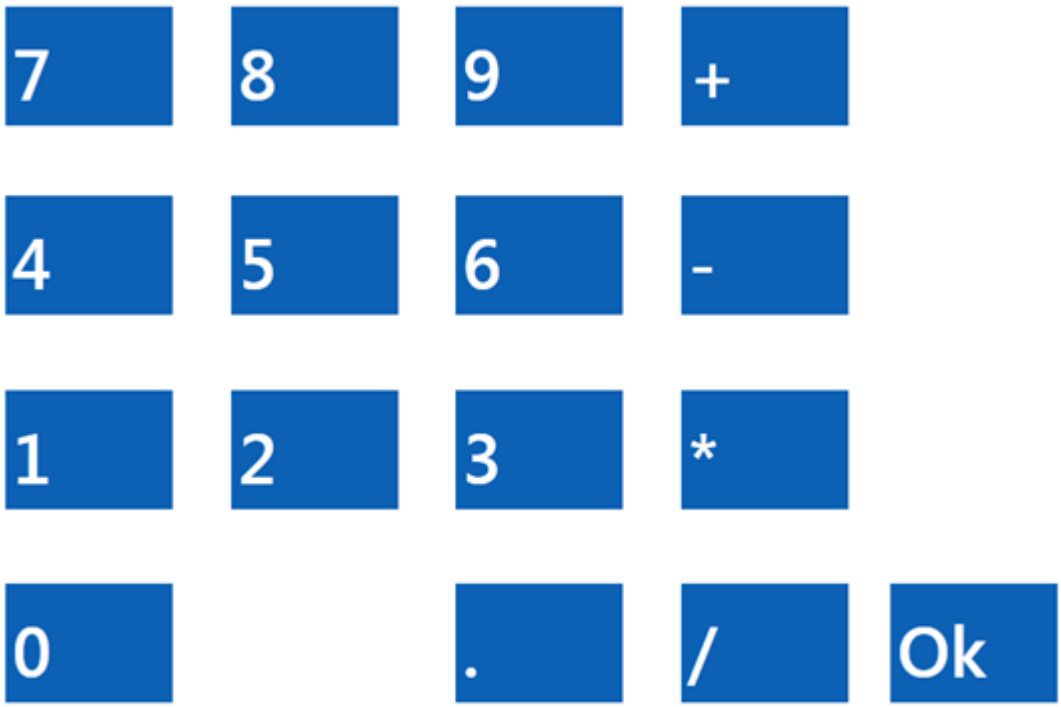


Figure 17. First pilot test layout

Our first pilot test was based on a calculator that we created. The calculator test was run for each of the 12 tested interaction types, and the calculations were tested in random order. The user had to press the ok button after entering each calculation. There were total of 26 button presses per one interaction type, including the calculations and the ok button press. We used five different calculations in our test:

- 1 + 3
- 5 * 6
- 6 / 3
- 888 – 777
- 9 + 8 + 7

The pilot testing indicated that the calculator-based test was too complex, since the focus should've been on the buttons, not on the calculations. The test was also found to be too lengthy, since some of the interaction types strained the participant. As we had expected, the first pilot testing results favored the two-handed push gesture. In addition, the push gesture was found to be the least efficient interaction type. Most importantly, however, the first pilot test helped us to evaluate our testing program, not the results themselves. By utilizing these findings, we had enough information to proceed to the second pilot test.

3.2.5. Second pilot test

For the second pilot test, the test setup was modified based on our findings from the first pilot testing. The calculation test setup was found to be too complicated for testing the buttons, and it was discarded. Our questionnaire was also modified to yield more accurate and relevant background information. The training screen instructions were simplified to better inform the participant about the next interaction type. Two of the interaction types, the two-handed sticky L-gesture button and the two-handed sticky push button, were also discarded. We felt that introducing two variants, the two-handed and the sticky, to both the push and L-gesture buttons at the same time was too much. The combination of both variants didn't offer any advantages over the single variants.

Like our first pilot test, we also based our second pilot test on numbered buttons, because number-based tasks are easy to complete. We changed the implementation to a 9-grid, following a generic keyboard numpad layout, and the nine numeric buttons were placed in the grid in the middle of the screen, leaving enough empty space around the testing zone and between the buttons.

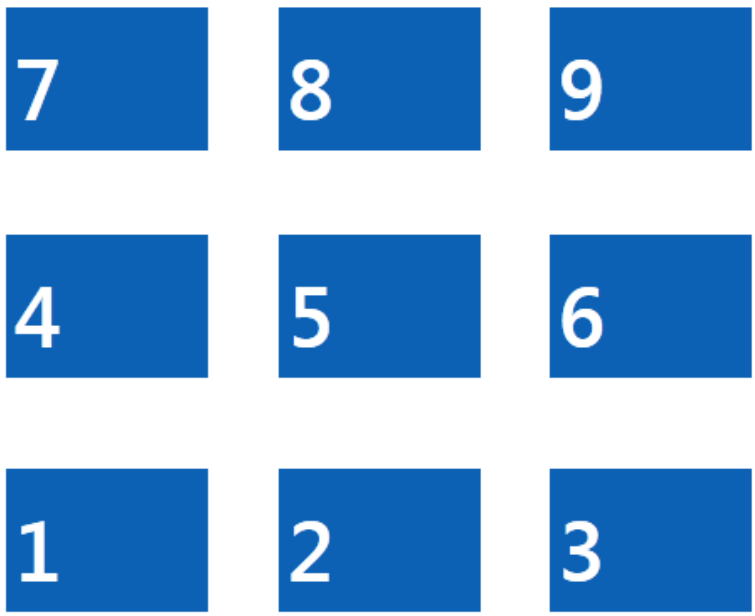


Figure 18. Second pilot test layout

There were a total of 18 button presses per one interaction type in the second pilot test. The tested numbers were selected randomly, so that the first six numbers were single digits. The remaining 12 numbers were also randomized using the same logic, but with the difference that all the numbers were dual digits of the same number, for example, the number 1 becoming 11. The double numbers were used to test repeated click actions for each of the interaction types. The following interaction types were included in the second pilot test:

- Hover button
- Confirm button
- Confirm hover button
- Gesture button
- Push button
- Sticky gesture button
- Sticky push button
- Two-handed gesture button
- Two-handed push button

Based on the second usability test, the layout was suitable to be used in the actual testing. Some of the interaction types, such as the two-handed L-gesture, were found to be unreliable, and required some additional developing. The results were similar to the first pilot testing and the results favored, too, the two-handed push gesture.

The final test setup was almost the same as the one used in the second pilot test. We updated the hand gesture recognition to be more accurate, the push gesture to register activations more easily, and fixed some minor bugs.

3.3. Programming

We used Microsoft Visual Studio 2010 for programming, and our programming language of choice was C#, since we were already proficient with it. The used framework was Windows Presentation Foundation (WPF). Our solution contained three Visual Studio projects. The first project was the main program containing all the algorithms, The Kinect device handling and some debugging tools. The second project was the usability test program containing the test user interfaces and recording tools. The third and final project was the test analysing program, which we used to transform recorded test data into a readable format. As calculated with the Visual Studio 2010 Code Metrics, our solution contains 2496 Lines of Code (LoC), with 1908 LoC in our main project.

The development work was done in cooperation with my colleague (M. O.). We divided the programming evenly, with he specializing in skeleton data handling, hand

gesture recognition and mouse control, and me specializing in the Kinect device programming, UI programming and depth data handling.

Some of the used imaging algorithms are from AForge Math Library [Kirillov, 2007]. AForge.NET framework is an open source C# framework related to computer vision and image processing. All of the AForge Framework algorithms were designed for 8-bit colour images, or they had some unnecessary code that slowed the computations down. Since the Kinect depth data is recorded in 16-bit, the algorithms were modified to work with the 16-bit data.

As the Kinect device provides us with 30 frames per second (fps) and the hand recognition needed to be executed every frame, the performance was always one of the most important design aspects during the programming. The hand gesture recognition algorithm took only about 1ms to 3ms per frame when running the program without the debug visualization. When the visualization was used, the program took about 90% of the CPU power, over 30ms per frame, and thus it was not used in our testing program.

The main application was programmed to be single thread. The threading would be the first place to look for performance improvements, but we felt that it wasn't necessary at the current level, since our recognition algorithm required less than 10% of the time available per frame.

3.3.1. Data from the Kinect device

There are three major types of data the Kinect sensor provides: depth data, skeleton data and RGB data. We chose not to use the RGB data, since we got all the information that we need from the depth data and skeleton data.

Skeleton data

Using the skeleton data the Kinect device provides is the easiest way to interact with NUIs. Multi-user NUIs can also be supported as the Kinect device can track the skeleton of two users and the number of identified users can be as high as six. The full skeletal structure of a user can be tracked when the user is fully visible to the Kinect camera, but the Kinect device also supports the seated mode, where only the upper torso is included.

There are many ways to select the active user in case there are many users in the Kinect device field of view area. My colleague (M. O.) developed an algorithm that we used to select the active Kinect user. In his algorithm, basically, only one user is tracked at a time. Once the user has been identified and tracked, another user cannot take controls until the tracked leaves the scene.

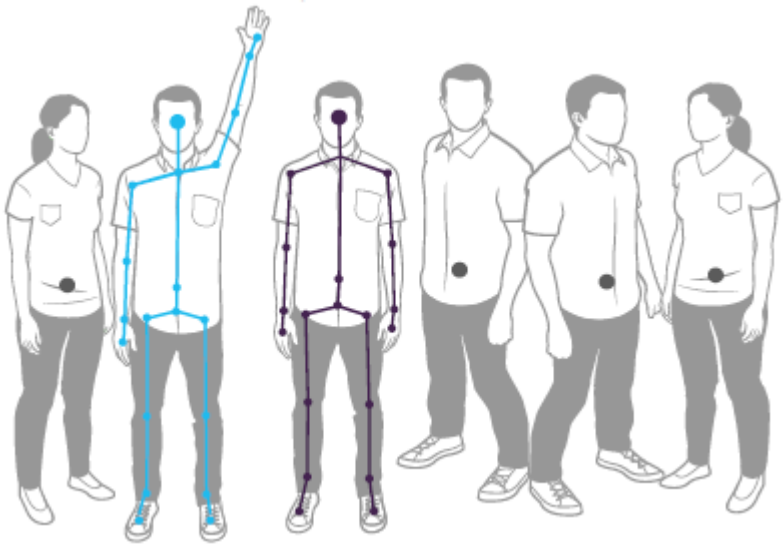


Figure 19. Two tracked skeletons [Microsoft, 2012]

The three of the most important skeletal points for us were the hand, the wrist and the shoulder points of the main arm. We could control the mouse and track one-handed gestures by using just these three points. The two-handed gestures required two additional skeletal points to be used, the hand and the wrist point of the other hand. To get the hand gesture recognition working, we also needed more data than merely the hand and wrist points. Unfortunately, the skeleton hand joint position isn't always accurate, which greatly affected our mouse controlling and hand gesture recognition accuracy.

The Kinect device offers some built-in smoothing parameters for the skeleton data, so that programmers can choose between speed and accuracy, whichever they need the most. These parameters are called *TransformSmoothParameters.* After careful evaluation, we chose the following parameters:

- Correction = 0
- JitterRadius = 0.15F
- MaxDeviationRadius = 0.15F
- Prediction = 0
- Smoothing = 0.3F

These parameters provided us more accurate cursor control than the Xbox 360 version. Jitter correction value was just at the level where the unwanted jitter was smoothed and at the same time the cursor remained responsive.

Depth data

The Kinect sensor offers depth data for developers in few different formats, the most useful being the type called *Resolution640x480Fps30*. This depth data type can also be described as the pixel resolution of 640 x 480 and the frame rate of 30 fps. The depth

data can be chosen to only include raw depth values or depth values with player index embedded into them. We chose to include the player index, since we used it in our calculations. If no player index was included, the depth data was returned as an array of short integer values, each individual short value representing a given distance from the Kinect sensor in millimeters. The same values can be calculated from the short integer values even if the player index is embedded.

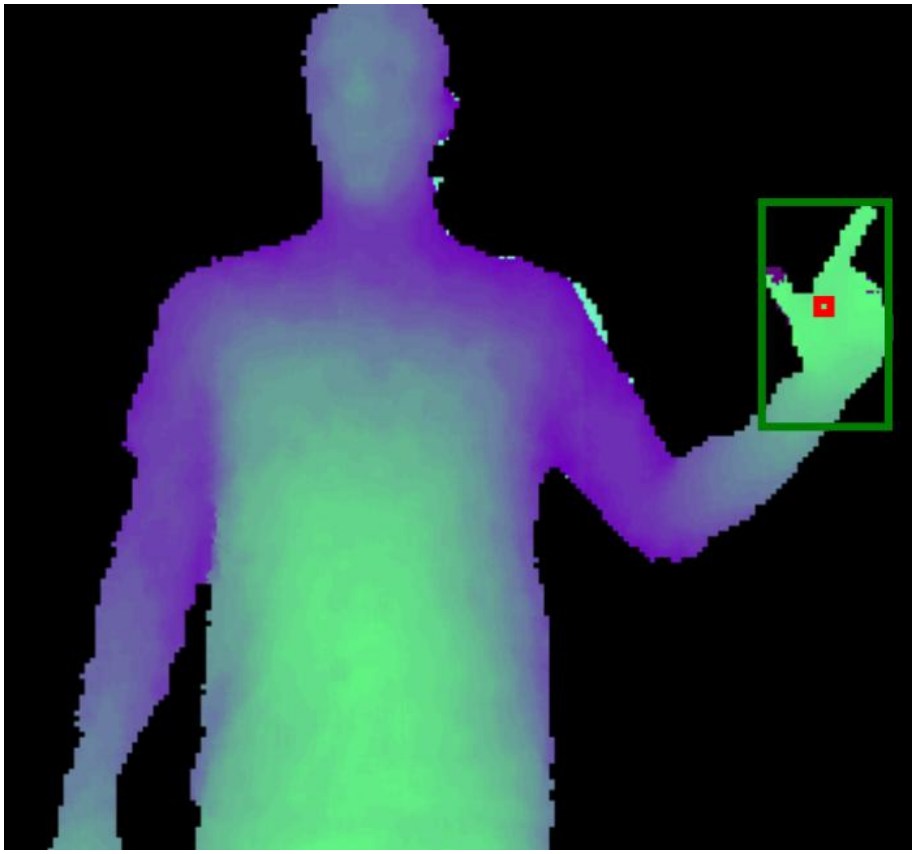


Figure 20. The depth visualization using only the player index data, including the hand skeleton point as the red rectangle and the hand area as the green rectangle

Unfortunately, the depth data contains fair amount of noise that cannot be avoided. Noise and poor data are something that is hard to fix without improvements to the hardware, but there were still some smoothing algorithms we could use. We tried several of those for the depth data, and after the evaluation of a few, we felt that the tradeoff between noisy and fast depth data and smooth and lagging depth data was too severe, so we chose not to use any smoothing for the depth data.

Since the Kinect device and the skeleton data don't offer us much to go on regarding the hand gesture recognition, we had to build the recognition ourselves, which we cover in detail in the next Section.

3.3.2. Mouse control and hand gesture recognition

There are a variety of ways how to control the cursor using the Kinect device. The most straightforward one is to map the Kinect skeleton hand position to the cursor position and scale it to reach the entire screen. We took this a step further and added the user shoulder point as the relative comparison point for the cursor manipulation. This way, the cursor stayed in the same place even if the user moved around, providing the hand-shoulder ratio stayed the same.

The hand gesture recognition was the most difficult and time-consuming part to implement. We reviewed several ways and algorithms to recognize hand gestures until we were satisfied with the recognition speed and accuracy [Wu and Huang, 1999].

Fortunately, we didn't have to start from scratch when implementing the hand gesture recognition. We had the approximate hand center point from the skeleton data. This point could be used as the starting point that is likely to be player's hand. We could get all pixels belonging to the user's hand by running an algorithm called queue-linear flood fill [Dunlap, 2006]. These pixels were then cropped to a smaller image, which was much faster to process than the whole image. The hand center point is then recalculated based on the hand area mass average. Using this recalculated hand center point together with the wrist center point from the skeleton data, we could approximate the hand rotation angle. The hand image was then rotated using this calculated angle to ensure that the hand image we use in the recognition was always in upright position. Finally, we resized the hand image to 20 x 30 pixels, so that our hand data was always uniform and easily comparable.

Figure 21. Final image of the hand before it was used for our recognition algorithm

There are many ways to recognize a hand gesture from this hand image. We tried several image recognition algorithms, but we felt that our own approach would still be most suitable. My colleague (M. O.) then developed his own hand gesture recognition algorithm that we used, which we called simplified keypoint matching algorithm. His algorithm was based on an algorithm called keypoint matching algorithm, and it could be simplified, because we had the hand image angle and the image size already calculated. Our algorithm had many advantages over the other reviewed algorithms, for example, it required very little CPU power.

We also kept the reliability in mind when we were developing and fine tuning the hand gesture recognition, since the recognition should work with different hand types. Our own algorithm had the highest recognition rate because it was easily configurable and modifiable during the iterative development.

3.4. Summary

In this Chapter, we discussed about our proposed new interaction types and how they were designed. We divided our new interaction types into three categories, button-based, gesture-based and two-handed variants, and then covered how they work and why they were chosen as our interaction types to be tested.

We developed a usability test to evaluate the interaction types. A pilot test was first run to see if our test setup was ready for testing. We confirmed our test setup validity using a second pilot testing, and after that was successfully completed, ran the usability test of 20 participants.

Lastly, we went for high-level look into what programming-related decisions we made during the development. We developed our new interaction types using C# as our chosen programming language, and the natural user interaction was done using the skeleton and depth data from the Kinect device. We also used the Kinect device for mouse control and hand gesture recognition.

In the next Chapter, we will address the results from the usability tests and see, whether they reveal any significant differences between the chosen interaction types. We will also see whether or not the tests confirm the hypotheses laid out earlier.

4. Results

In this Chapter we will go for an in-depth review of our results and findings. First we will process the main results and go through the statistics. We will compare the original interaction types to the new proposed interaction types to see if there are any significant differences in the error rates and completion times. We will also compare the one-handed interaction types with the two-handed variants.

The completion time of interaction types and the false button activations were the primary data for our research. The secondary data of the research consisted of the missed button activations and the usability test participant observations.

The missed button activation results are covered in the second part of our results, since it was not possible to reliably measure missed activations with all of our interaction types. For example, with the hover button, a missed activation is when the cursor leaves the button area before the button is activated. However, we cannot differentiate whether the cursor leaves the button area by accident or voluntarily.

4.1. Statistical methods

We used Fisher's exact test [Fisher, 1922] to calculate significant differences between the correct and the false button activation amounts. The completion times were analyzed using the two-tailed t-test [Freund, 1984] for repeated observations to calculate significant completion time differences between each of the interaction types. Finally, we also observed when the participants noticeably expressed positive or negative reactions.

4.2. Main results

All usability tests were recorded using the test recoding program that we developed. The recorded data contained:

- Completion time of each of the interaction types
- Correct and false button activations
- Missed button activations
- Individual click times and coordinates

The individual click times and coordinates were not used in our main results.

For the completion times and false button activations, we obtained statistically significant data, which enabled us to leave studying individual clicks out of this research. The mouse training data was also recorded but it was not included in the

results. Since it was the first test and it was only used to familiarize the participants to the UI and the test layout, it was not comparable to the other interaction types.

To keep the statistics in an easily readable format, the following abbreviations are used in the results:

- **H** Hover button
- **CH** Confirm hover button
- **2P** Two-handed push button
- **2G** Two-handed gesture button
- **G** Gesture button
- **P** Push button
- **C** Confirm button
- **SP** Sticky push button
- **SG** Sticky gesture button

4.2.1. False button activations

The difference between the correct and the false button activation amounts are later referenced as *the error rate*.

	H	CH	2P	2G	G	P	C	SP	SG
Unusable values	0	0	0	0	3	0	0	0	1
No false activations	19	20	18	14	11	14	11	7	9
One or more false activations	1	0	2	6	6	6	9	13	10
Two or more false activations	1	0	0	2	1	3	5	6	4
Three or more false activations	1	0	0	0	1	1	2	3	3

Table 5. The error rates

There were four unusable values as the participants didn't complete the tests; our algorithm was not able to recognize the L-gesture in reasonable time. Three of those were from the L-gesture button test and one was from the sticky L-gesture button test.

There are four different abbreviations used in the following table: +++ where $p < 0.001$; ++ where $p < 0.01$; + where $p < 0.05$, and n.s. marking for non-significant values. The + values in the table are read horizontally, meaning that the row interaction type is significantly better than the column interaction type.

	CH	2P	2G	G	P	C	SP	SG
H	n.s.	n.s.	n.s.	n.s.	n.s.	++	+++	++
CH		n.s.	+	+	+	++	+++	+++
2P			n.s.	n.s.	n.s.	+	+++	+
2G				n.s.	n.s.	n.s.	n.s.	n.s.
G					n.s.	n.s.	n.s.	n.s.
P						n.s.	n.s.	n.s.
C							n.s.	n.s.
SG								n.s.

Table 6. The significant differences between the error rates

The amount of *one or more false button activations* was compared to the amount of *no false button activations*, which are shown the table (table 5). Based on the error rate results, the confirm hover button had significantly smaller error rate (where $p < 0.05$) than all the other interaction types, except for the two-handed push button or the hover button. There was no statistical difference in the error rate between the two-handed push button and the hover button, nor there was between the two-handed push button and the push button.

We assumed that introducing the sticky variant would significantly reduce the error rate. However, the confirm button did have significantly larger error rate than the confirm hover button ($p < 0.01$).

The error rates are presented in the graph (fig. 22). The amount of clicks per interaction type is quite low (N = 20), but the results should still give a reasonably good illustration of the false button activation rate correlation between the interaction types.

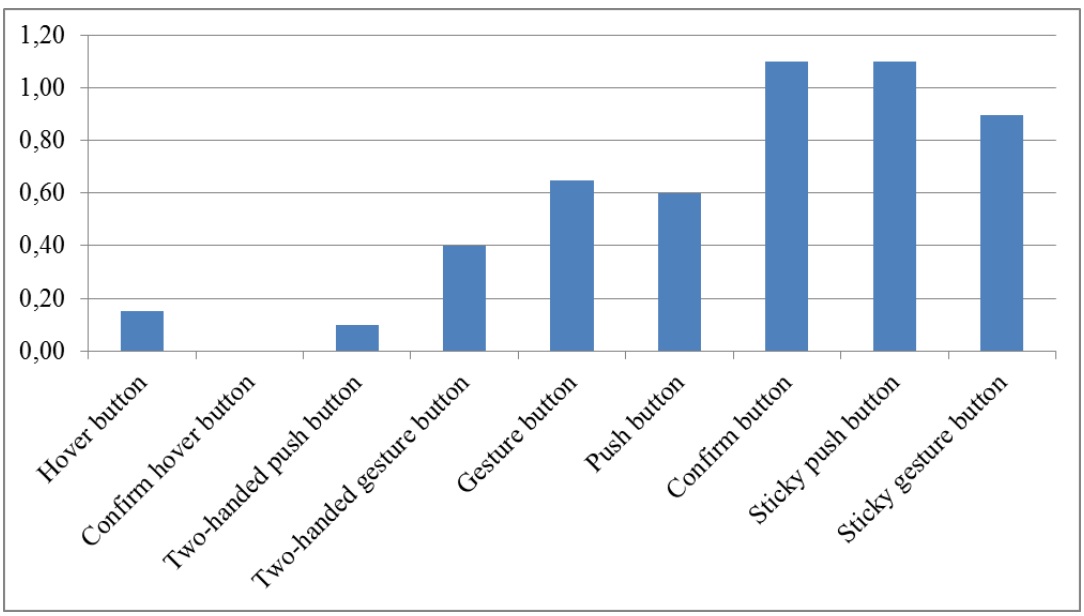


Figure 22. Mean error rate per tested interaction type

4.2.2. Completion times

We measured the time it took to complete each of the interaction type tests. Seven different abbreviations were used in the following table (table 7) to show significant differences between the interaction types: +++ where $p < 0.001$; ++ where $p < 0.01$; + where $p < 0.05$; – – – where $p < 0.001$; – – where $p < 0.01$; – where $p < 0.05$, and n.s. marking non-significant values. The + values in the table are read horizontally, meaning that the row interaction type is significantly better than the column interaction type. On the other hand, the – values in the table are read vertically, meaning that the column interaction type is significantly better than the row interaction type.

	CH	**2P**	**2G**	**G**	**P**	**C**	**SP**	**SG**
H	+++	– – –	n.s.	n.s.	n.s.	– – –	– – –	n.s.
CH		– – –	–	n.s.	– –	– – –	– – –	n.s.
2P			++	++	+++	+++	+++	+++
2G				n.s.	n.s.	n.s.	n.s.	n.s.
G					n.s.	–	n.s.	n.s.
P						– – –	– – –	n.s.
C							n.s.	++
SP								++

Table 7. Significant differences between the completion times

The following table (table 8) contains calculated values for the completion times. For each of the interaction types, the mean completion times, the standard deviations, and the minimum and maximum values were calculated.

	Mean	**SD**	**Min**	**Max**
H	0:00:56,500	0:00:05,969	0:00:48,000	0:01:09,000
CH	0:01:12,500	0:00:11,700	0:00:54,000	0:01:47,000
2P	0:00:32,850	0:00:06,777	0:00:20,000	0:00:44,000
2G	0:00:55,300	0:00:30,802	0:00:24,000	0:02:15,000
G	0:01:12,529	0:00:52,913	0:00:32,000	0:03:39,000
P	0:01:01,550	0:00:18,594	0:00:54,000	0:01:50,000
C	0:00:47,100	0:00:10,052	0:00:39,000	0:01:04,000
SP	0:00:45,850	0:00:11,609	0:00:28,000	0:01:16,000
SG	0:01:03,105	0:00:21,561	0:00:37,000	0:01:38,000

Table 8. The completion time values

The mean completion times for each of the interaction types are presented using the blue bars in the graph (fig. 23) below. The standard deviation (SD) of the completion time is presented using the black vertical lines.

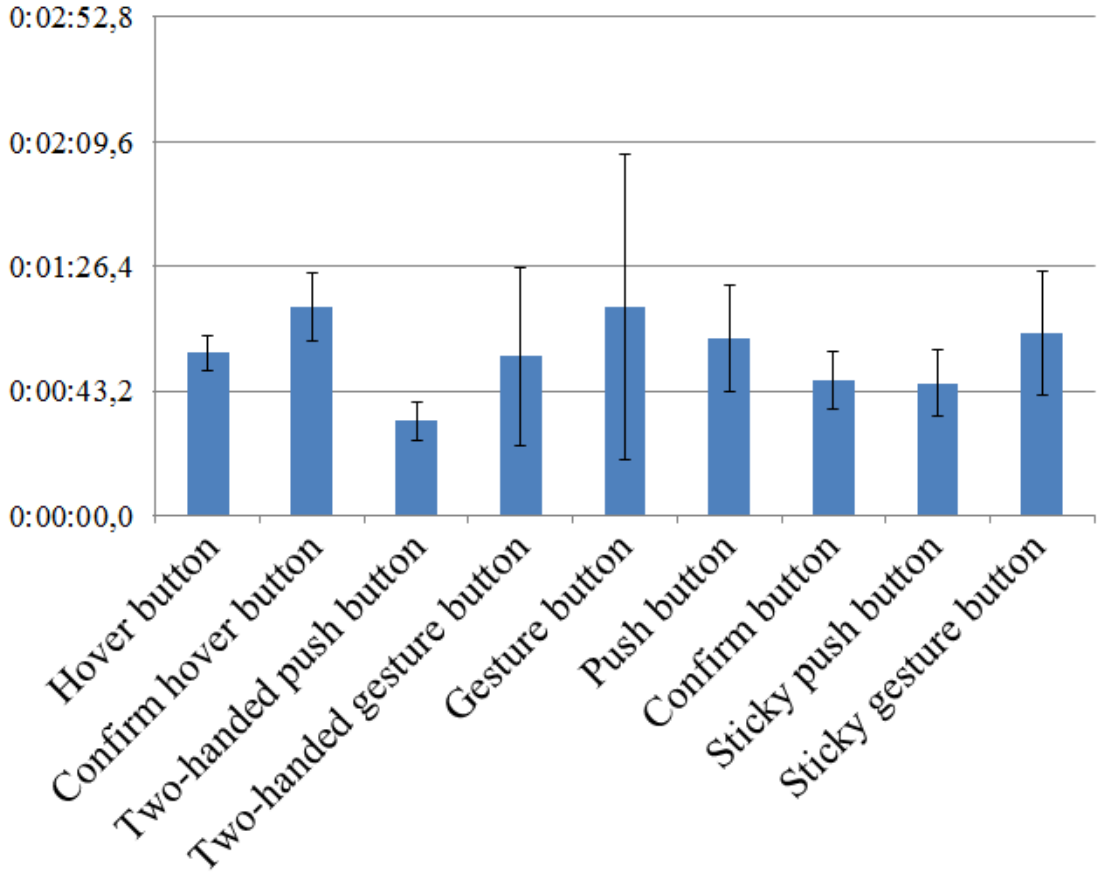


Figure 23. The completion times for different interaction types

All in all, the two-handed push button was significantly faster than all the other interaction types ($p < 0.05$). The sticky push button was significantly faster than the push button ($p < 0.001$). Introducing the sticky variant to the L-gesture button didn't have any statistical significance concerning the completion. There was no significant difference in the completion times of the two-handed L-gesture button and the L-gesture button

4.2.3. Original and proposed new interaction types

Hover button

The completion time of the hover button was compared to the completion times of the proposed new interaction types. There were no significant differences in the completion times between the hover button and any of the L-gesture-based buttons or the push button. However, the two-handed push button, sticky push button and confirm button were significantly faster than the hover button ($p < 0.001$).

Comparing the error rate of the hover button to the error rates of the proposed new interaction types, only the two-handed push button had no significant difference.

Confirm hover button

The confirm hover button was the other original user interaction type we compared with the proposed new interaction types. Only the L-gesture button and the sticky L-gesture button had no statistical difference in completion times compared to the confirm hover button. The rest of our proposed new interaction types were significantly faster ($p < 0.05$) than the confirm hover button.

The confirm button was significantly faster ($p < 0.001$) than the confirm hover button. However, the error rate for the confirm button was far greater ($p < 0.01$) than in the confirm hover button. In addition, the confirm hover button had the least amount of false button activations.

4.2.4. One-handed and two-handed interaction types

The two-handed push button was significantly faster ($p < 0.001$) than the push button, but there was no statistical difference between the error rates of these buttons. As a matter of fact, the two-handed push button was the fastest of our interaction types and at the same time had nearly zero false button activations.

There was no significant difference between the completion times of the L-gesture button and the two-handed L-gesture button, nor was there difference in the error rates.

4.3. Secondary results

As secondary results, we used the data collected from the observation of the participants and their post-test interviews to complement our main results.

4.3.1. Missed button activations

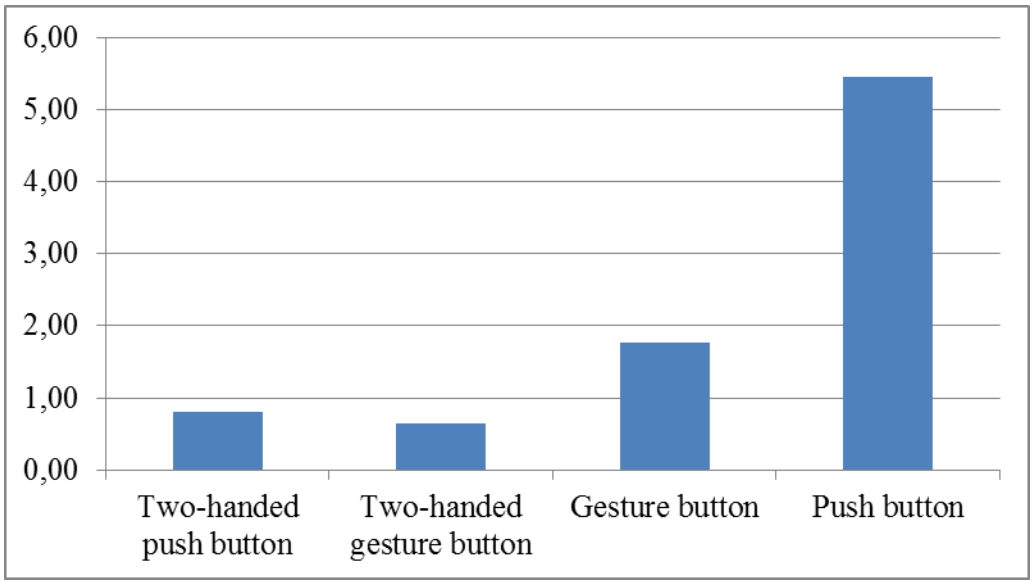
There are four different abbreviations used in the following table: +++ where $p < 0.001$; ++ where $p < 0.01$; + where $p < 0.05$, and n.s. marking for non-significant values. The + values in the table are read horizontally, meaning that the row interaction type is significantly better than the column interaction type.

	2P	**2G**	**G**	**P**
2P		n.s.	n.s.	+++
2G			n.s.	+++
G				+
P				

Table 9. Significant differences between the missed button activations

Missed button activations were only possible with the two-handed push button, two-handed L-gesture button, L-gesture button and the push button. Thus, the amounts of missed button activations are only comparable between these four interaction types.

Only the push button produced significantly more ($p < 0.05$) missed button activations than the other interaction types. When compared to the two-handed push button, the amount of missed button activations was extremely significant ($p < 0.001$).



Graph 3. Mean missed button activation rate per tested interaction type

We compared the missed button activation rates of the L-gesture button and the two-handed L-gesture button, and found no statistically significant difference. However, when interpreting the graph 3, we can see a tendency of the two-handed L-gesture button having less missed button activations than the L-gesture button.

4.3.2. Usability test participant observations and comments

The hand shape of an individual participant varies greatly which affected our recognition rate severely, since the hand gesture recognition was configured using the hand of the other researcher (M. O.). We could have obtained different results had the hand recognition been configured individually for each of the participants. Especially female participants' hands were considerably smaller than our preset configuration, which reduced the gesture recognition accuracy.

One of the participants expressed a wish to able to relocate the cursor by using relative coordinates similar to how the mouse works. Making the cursor work like a normal mouse controlled cursor via the Kinect device is technically possible, and we had such implementation at one stage of our development. However, we still chose to use cursor control based on absolute coordinates, since only the gesture-based interaction types worked with the relative coordinates.

Some interaction types, such as the hover button and the confirm hover button required the user to hold the hand still for periods of time, which few of them found difficult. The rest of our interaction types were more dynamic and did not require the hand to be held at the same place for relatively long periods of time. Instead, the participants could choose when to activate the buttons. This way, the participants could set the pace to their liking.

We collected detailed observations regarding the sticky variants. Two of the participants found the sticky variants to be more functional and comfortable than the non-sticky versions. In contrast, one participant had trouble learning the sticky concept.

The L-gesture variants were observed in general, and one of the participants remarked the L-gestures as the most functional. However, five of the participants felt the L-gestures to be the least functional. Two of the participants had even trouble learning the L-gesture, and one participant had trouble aiming the L-gesture towards the Kinect device.

We asked the participants to choose their favorite and least favorite of the interaction types. Even though there were only 20 participants, there were 23 favourites. The higher number can be explained by the fact that some of the participants could not decide between two of their favorite interaction types.

	Favorite	**Least favorite**
Hover button	3	3
Confirm hover button	0	1
Two-handed push button	13	0
Two-handed gesture button	3	1
Gesture button	2	7
Push button	1	7
Confirm button	1	1
Sticky push button	0	0
Sticky gesture button	0	0

Table 10. Participants' preferences

The two-handed push button received over 50% of the most favorite votes. This is encouraging to our research, since the users' opinions are greatly valued factor in the field of usability. Even if an interaction type is fast to use and error-free, it can still be uncomfortable to use. The users are more likely to choose a comfortable and slow interaction type over a fast and uncomfortable one.

The participants had mixed feelings about the hover button. Three of them ranked it as their favorite and another three as their least favorite interaction type. The participants favoring the hover button were fond of the fact that the button was activated

for them, whereas the participants disliking the hover button felt that the button activation was out of their hands and at the same time too slow. Further research would be required to make conclusions whether there is a difference between how gamer users and casual users experience the interaction speed of the hover button.

	Negative	Positive
Hover button	0	0
Confirm hover button	2	0
Two-handed push button	0	1
Two-handed gesture button	3	0
Gesture button	1	1
Push button	4	0
Confirm button	2	0
Sticky push button	2	0
Sticky gesture button	3	0

Table 11. Observation of the participants' feelings

We observed the participants during the usability tests and noted when their expression was notably positive or negative. All of our new interaction types, except the two-handed push button, received negative comments from the participants. These negative reactions can mostly be explained by the unreliable performance of the L-gesture variants as well as keeping the one-handed push button in the usability tests even if it was not suitable for comfortable user interaction.

Only the hover button and the two-handed push button didn't cause negative reactions in the participants. In fact, the two-handed push button and the L-gesture button were the only ones that raised positive reaction in the participants.

4.4. Summary

In this chapter, we laid out our results from the usability tests. Based on the results, the two-handed push button was significantly faster than other interaction types. Removing the hover part from the confirm hover button made it significantly faster but at the same time raised the error rate significantly higher. The L-gesture-based variants performed unreliably, and the participants generally disliked them.

The results were in favour of our main interaction type, the two-handed push button, which we will discuss it in more detail in the next Chapter. We will also compare how our other proposed interaction types fared against the original interaction types. Lastly, we will discuss about future research topics, and look how our proposed interaction types could improve existing Kinect-based games.

5. Discussion

In this Chapter we reflect our findings and discuss how the proposed new interaction types can be incorporated into real world use.

The data from the false button activations was one of our most important data sources. We used the false button activation amounts to find out if there were any statistical differences between the interaction types. As for the missed button activations, we did not use them in our main results, since they tend not to be as critical part of the UI interaction as the false button activations.

However, missing a button or a target can be considered critical in a gaming environment. For example, a missed shot in a first person shooter can mean a difference between the life and death of your character. Thus, significant missed button activations should still be taken into consideration when choosing interaction types for games that require precision and speed.

As mentioned earlier, in our testing, the missed button activations could only be reliably measured in the two-handed push button, two-handed L-gesture button, L-gesture button and the push button. In theory, other interaction types could also produce missed button activations. For example, in missed button activation of the hover button, the user is holding the cursor over the button but then accidentally leaves the button area. However, this interpretation is not optimal, since it does not differentiate between a missed button activation and an intentional cancellation of the button activation.

Only the hover button, confirm hover button and the two-handed push button had small enough error rate to be used for reliable user interaction in their current development state. Further development of the rest of the interaction types could drastically lower the error rates, and could potentially make them viable alternatives for NUI user interaction.

According to the HIG document [Microsoft, 2012], the interaction speed is one of the factors that should not be focused on when designing NUIs. Contrary to this view, we felt that the interaction speed is one of the most valuable attributes of fluent user interaction. However, the interaction speed in itself is not sufficient, and we need to take the error rate into account as well.

We compared the confirm button to the confirm hover button to see if removing the hover part would speed up the interaction speed and keep the error rate at an acceptable level. The fact that the error rate in the confirm button was far greater than in the confirm hover button led to the conclusion that removing the hover button behavior from the confirmation hover button was not a viable option, and that the confirm button should not be used in NUIs.

In theory, the two-handed L-gesture button should be as fast as the two-handed push button. Unfortunately, our recognition algorithm worked unreliably for different hand types, which mostly explains the large standard deviation of the L-gesture-based interaction types and the poor performance of the L-gesture-based buttons.

Introducing the sticky variant to the L-gesture button did not have any statistical significance concerning the completion times, which did not confirm our hypothesis. We had thought that the sticky L-gesture button would have faster completion time than the non-sticky version of the L-gesture button. This, however, was the case when adding the sticky variant to the push button, as it did improve the completion time significantly.

Against our hypothesis, introducing the sticky variant to the push button or the L-gesture button did not have significant effect on the error rates. In addition, it is noteworthy that the push button had significantly more missed button activations than the two-handed push button. This led us to the conclusion that the slow interaction speed and the high false button activation amount renders the push button unsuitable for comfortable NUI interaction, at least in the application environment. However, it can provide just enough challenge when used in a forgiving gaming environment.

Changing the push gesture from one-handed to two-handed version made a large difference in terms of interaction speed, and at the same time reduced the amount of false button activations. This finding was highly positive, and led us to the conclusion that the two-handed push button can and should be used in NUIs to complement or replace current interaction types. Our findings also suggest that the two-handed push gesture is accurate and fast enough to be used in advanced gaming environments, such as first person shooters; the Kinect device could be used for fast and comfortable user interaction.

Even the most recent Kinect-based games, such as the Fable: The Journey [Lionhead, 2012], still use the one-handed push gesture for various interaction types. As we mentioned earlier, the one-handed push gesture can be great fun in a gaming environment, but the inaccuracy can be too much for some users. In his review of Fable, Lang [2012] criticizes the controls not being perfect, and states that such an issue would be unforgivable in a shooting game. We propose that Fable would also benefit from two-handed push gesture, but it could turn out that it would be too accurate and the game would be too easy in its current state.

In its current state, the two-handed push button is suitable for different kinds of natural user interactions, such as controlling the mouse in the Windows environment. However, although the accuracy can always be fine-tuned, there is not much room for further development. On the other hand, the L-gesture variants could benefit greatly from future development. There are currently no games on the market that use hand gestures for user interaction, partly due to the difficulty of recognizing hand gestures reliably, partly due to lower resolution of the Xbox version of the Kinect device. Our

work could provide the necessary push for the developers to come up with more ways to utilize hand gestures in games as the means of more natural user control.

For the conclusion, based on the statistics and the user feedback, we found our proposed new interaction type, the two-handed push button, to be the most usable NUI button interaction type with the Kinect device. The low error rate of the two-handed push button combined with the fast interaction speed makes it the only proposed new interaction type that we can recommend.

6. Conclusion

We wanted to improve the usability of the Kinect-based NUI as soon as it was available for the consumers. We weren't content with how long the UI interaction took, so we started to consider other possibilities. After reviewing the existing Kinect-based interaction types, we found the swipe button the most comfortable and fastest to use. Unfortunately, the swipe button requires a very special UI and is not suitable for cursor-based environments, since it cannot be used with the cursor. This very limitation pushed us to develop our own alternatives.

After familiarizing ourselves with the Kinect device, we developed our own algorithm for controlling the cursor. This cursor control was a vital step in our progress; because all of our interaction types rely on the cursor, it had to be developed and fine-tuned first. With working cursor control, we could finally develop prototypes for our new interaction types. The development of the Kinect-based new interaction types proved to be quite a challenge, especially so with the interaction types based on hand gestures.

Removing the hover button functionality from the confirmation hover button was not a viable option in terms of reliable usability. The confirmation hover button was significantly slower than the hover button, and they did not have a significant difference in the error rate. This finding confirmed that the confirmation hover button is too slow for normal user interaction, and should only be used in special cases, where any mistakes should absolutely not happen.

The two-handed push button was significantly faster than the hover button, and at the same time had no significant difference in the false button activation rate. It was also the interaction type the participants preferred. This significant discovery should encourage people developing Kinect-based NUIs to try our proposition, the two-handed push button, as an alternative to popular NUI button types, such as the hover button.

Individual click coordinates and times were not used in our research even though they were recorded. Additional statistical analysis could be done using the data already collected. We could compare the click times for each interaction type comparing only the double digit part of the data. Another angle from which to interpret our data would be how far off the individual clicks are from the button center.

Some of our proposed new interaction types may have limitations and can definitely be developed further. Careful consideration should still be used when choosing the interaction types for further development, since some of the proposed new interaction types didn't yield very positive results.

One of the most important findings was that the two-handed push button is a viable alternative for interacting with NUI buttons. With the ever-evolving technology, eventually the most subtle body gestures can be taken into account when designing new interaction types. The two-handed push gesture is just the first step of using the whole human body for a reliable user interaction. Someday, we can use facial expressions and subtle finger gestures to control NUIs from anywhere, not just when facing a stationary sensor such as the Kinect device.

References

[Bruegger and Hirsbrunner, 2009] Pascal Bruegger and Béat Hirsbrunner, Kinetic user interface: interaction through motion for pervasive computing systems. In: Constantine Stephanidis (ed.), *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments.* Springer, 2009, 297-306. doi: 10.1007/978-3-642-02710-9_33

[Dix, 2002] Alan Dix, *Incidental Interaction.* 2002. Retrieved from http://alandix.com/academic/topics/incidental/dix-incidental2002.pdf (27.11.2012)

[Dunlap, 2006] Justin Dunlap, *Queue-Linear Flood Fill: A Fast Flood Fill Algorithm. 2006.* Retrieved from http://www.codeproject.com/Articles/16405/Queue-Linear-Flood-Fill-A-Fast-Flood-Fill-Algorith (27.11.2012)

[Fisher, 1922] Ronald A. Fisher, On the interpretation of $\chi 2$ from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society* **85** (1), 1922, 87–94. doi:10.2307/2340521

[Freund, 184] John E. Freund, *Modern Elementary Statistics, sixth edition.* Prentice-Hall, 1984.

[Freeman, 2010] Will Freeman, *Should Kinect UI be Standardised?* 2010. Retrieved from http://www.develop-online.net/news/36304/Should-Kinect-UI-be-standardised (27.11.2012)

[Harmonix, 2010] Harmonix, *Dance Central*. 2010. Retrieved from http://www.dancecentral.com/ (27.11.2012)

[Holmes, 2011] Chris Holmes, *Kinect and the Future of Gestural Interfaces*. 2011. Retrieved from http://notesondigital.com/2011/06/kinect-and-the-future-of-gestural-interfaces/ (27.11.2012)

[Kieras, 2001] David Kieras, *Using the Keystroke-Level Model to Estimate Execution Times.* 2001. Retrieved from http://www-personal.umich.edu/~itm/688/KierasKLMTutorial2001.pdf (27.11.2012)

[Kirillov, 2007] Andrew Kirillov, *AForge.NET Framework.* 2007. Retrieved from http://www.aforgenet.com/ (27.11.2012)

[Kortum, 2008] Philip Kortum, *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory and Other Nontraditional Interfaces*. Morgan Kaufmann, 2008.

[Lang, 2012] Derrik J. Lang, Review: 'Fable: The Journey' casts Kinect magic. 2012. Retrieved from http://www.salon.com/2012/10/09/review_fable_the_journey_casts_kinect_magic/ (21.12.2012)

[Lewis and Rieman, 1994] Clayton Lewis and John Rieman, *Task-Centered User Interface Design - A Practical Introduction.* 1994. Retrieved from http://hcibib.org/tcuid/ (27.11.2012)

[Lionhead, 2012] Lionhead Studios, Fable: The Journey. 2012. Retrieved from http://www.xbox.com/fable (21.12.2012)

[Mann, 1998] Steve Mann, Reconfigured self as basis for humanistic intelligence. In: *Proc. of the annual conference on USENIX Annual Technical Conference USENIX Association Berkeley ATEC'98*, 15-19.

[Microsoft, 2012] Microsoft, *Human Interface Guidelines. Kinect for Windows v1.5.0.* 2012. Retrieved from http://go.microsoft.com/fwlink/?LinkID=247735 (27.11.2012)

[Nielsen, 2006] Jakob Nielsen, *Quantitative Studies: How Many Users to Test?* 2006. Retrieved from http://www.useit.com/alertbox/quantitative_testing.html (27.11.2012)

[Nielsen, 2010] Jakob Nielsen, *Kinect Gestural UI: First Impressions.* 2008. Retrieved from http://www.useit.com/alertbox/kinect-gesture-ux.html (27.11.2012)

[Nitsch, 2011] Peter Nitsch, *Insights into Kinect UI.* 2011. Retrieved from http://www.teehanlax.com/labs/insights-into-kinect-ui/ (27.11.2012)

[Normand and Nielsen, 2010] Donald A. Norman and Jakob Nielsen, Gestural interfaces: a step backward in usability. *ACM CHI* **17** (5), 2010, 46-49. doi: 10.1145/1836216.1836228

[NUI Group, 2009] NUI Group Community FAQs, *What is a "natural user interface"?* 2009. Retrieved from http://nuigroup.com/faq/ (27.11.2012)

[Nutt, 2010] Christian Nutt, *MIGS 2010: Harmonix's Solution for Kinect UI Design.* 2010. Retrieved from http://www.gamasutra.com/view/news/31414/MIGS_2010_Harmonixs_Solution_For_Kinect_UI_Design.php (27.11.2012)

[Iwamoto, 2008] Takayuki Iwamoto, Mari Tatezono, Takayuki Hoshi, Hiroyuki Shinoda, *Airborne Ultrasound Tactile Display*. SIGGRAPH 2008 New Tech Demos, 2008. Retrieved from http://www.alab.t.u-tokyo.ac.jp/~siggraph/09/TouchableHolography/SIGGRAPH08_abst.pdf (27.11.2012)

[Sova and Nielsen, 2003] Deborah H. Sova and Jakob Nielsen, *234 Tips and Tricks for Recruiting Users as Participants in Usability Studies.* 2003. Retrieved from http://www.nngroup.com/reports/tips/recruiting/234_recruiting_tips.pdf (27.11.2012)

[Ubisoft, 2010] Ubisoft, *Your Shape: Fitness Evolved*. 2010. Retrieved from http://yourshapegame.ubi.com/fitness-evolved-2012/en-us/index.aspx (27.11.2012)

[Vertegaal and Poupyrev, 2008] Roel Vertegaal and Ivan Poupyrev, Organic user interfaces: introduction. In: *Communications of the ACM 51* **6**, 26-30. doi: 10.1145/1349026.1349033

[Vredenburg *et al.*, 2002] Karel Vredenburg, Ji-Ye Mao, Paul W. Smith and Tom Carey, A survey of user-centered design practice. In: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems CHI'02*, 471-478. doi: 10.1145/503376.503460

[Wu and Huang, 1999] Ying Wu and Thomas S. Huang, Vision-based gesture recognition: A review. In *Proceedings of the International Gesture Workshop on Gesture-Based Communication in Human-Computer Interaction* (GW '99), 103-115.

Appendix 1: Usability test consent form

Please read and sign this form.

In this usability test:

- You will be asked to fill in a questionnaire.
  .
- You will be asked to perform certain tasks using Kinect.
- We will also conduct interview with you after the test.

Participation in this usability study is voluntary. All information will remain strictly confidential. The descriptions and findings may be used in our master's thesis. However, at no time will your name or any other identification be used. You can withdraw your consent to the experiment and stop participation at any time.

If you have any questions after today, please contact Tommi Pirttiniemi at tommi.pirttiniemi@uta.fi or 040-7067366

I have read and understood the information on this form and had all of my questions answered

_____

Subject's Signature

_____ _____

Usability Consultant Date

Appendix 2: Questionnaire

Purpose:

How well participants can interact with different buttons using only Microsoft Kinect as their input device.

Introductory Questions

- Have your ever used Kinect? ____Often ____Few times
  ____Once ____Never

- Select your age group ____18 to 25 ____26 to 35
  ____36 to 50 ____ over 50

- Do you have any handicap or disability that might affect your arm movements? ____Yes ____No

- How would you rank your computer related skills? ____Beginner
  ____Intermediate
  ____Expert

Appendix 3: Tested interaction types

- **H** Hover button
- **CH** Confirm hover button
- **2P** Two-handed push button
- **2G** Two-handed gesture button
- **G** Gesture button
- **P** Push button
- **C** Confirm button
- **SP** Sticky push button
- **SG** Sticky gesture button

Appendix 4: Completion times of the interaction types

HoverButton	ConfirmHoverButton	TwoHandedPushButton	TwoHandedGestureButton	GestureButton	PushButton	ConfirmButton	StickyPushButton	StickyGestureButton
0:00:57	0:01:07	0:00:33	0:00:34	0:00:32	0:00:47	0:00:42	0:00:43	0:00:41
0:00:52	0:01:06	0:00:23	0:02:15	0:00:37	0:00:51	0:00:42	0:00:53	0:00:47
0:00:53	0:01:07	0:00:32	0:00:53	0:00:41	0:01:06	0:00:45	0:00:35	0:01:07
0:00:54	0:01:13	0:00:43	0:00:28	0:03:09	0:00:58	0:00:36	0:00:34	0:01:15
0:00:57	0:01:09	0:00:28	0:00:38	0:00:46	0:00:47	0:00:46	0:00:38	0:00:38
0:00:49	0:01:00	0:00:27	0:00:41	0:00:40	0:00:53	0:00:38	0:00:34	0:00:41
0:01:01	0:01:16	0:00:37	0:00:55	0:03:39	0:00:49	0:01:00	0:00:42	0:01:11
0:00:51	0:01:07	0:00:28	0:01:09	0:00:37	0:00:39	0:00:41	0:00:45	0:00:37
0:01:09	0:01:47	0:00:44	0:01:05	0:00:54	0:01:50	0:01:03	0:00:45	0:00:44
0:00:55	0:01:10	0:00:35	0:00:38	0:00:41	0:01:17	0:00:50	0:01:01	0:01:38
0:00:55	0:01:31	0:00:43	0:00:36		0:01:29	0:01:01	0:00:57	0:01:35
0:01:02	0:01:08	0:00:35	0:00:53		0:01:10	0:00:53	0:00:45	0:01:37
0:01:02	0:01:26	0:00:36	0:00:45		0:00:59	0:00:57	0:00:52	
0:01:00	0:01:12	0:00:32	0:02:13	0:01:31	0:00:47	0:00:44	0:00:48	0:00:58
0:00:58	0:01:07	0:00:29	0:00:56	0:01:22	0:01:10	0:00:31	0:00:51	0:01:26
0:00:50	0:01:07	0:00:33	0:00:42	0:01:05	0:01:07	0:00:54	0:01:16	0:00:50
0:00:48	0:00:54	0:00:25	0:00:24	0:00:49	0:00:54	0:00:36	0:00:32	0:00:48
0:01:08	0:01:20	0:00:42	0:01:27	0:01:24	0:01:30	0:01:04	0:00:58	0:00:52
0:00:50	0:01:05	0:00:20	0:00:33	0:01:12	0:00:39	0:00:36	0:00:28	0:01:31
0:00:59	0:01:18	0:00:32	0:00:41	0:00:54	0:00:49	0:00:43	0:00:40	0:01:03