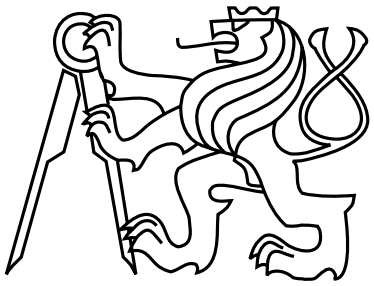
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Mid-Air Text Entry Methods

Barbariyskiy Nikolay

Supervisor: Ing. Poláček Ondřej

Study Programme: Open Informatics

Field of Study: Software engineering

May 25, 2014

Aknowledgements

First and foremost, I would like to thank to the supervisor of this thesis Ing. Poláček Ondřej, for the valuable guidance and advice and for showing me some example that related to the topic of this project. He inspired my greatly part of this project. Also, I would like to take this opportunity to thank to Tatiana Tarasova for her support and help in completing this project. Besides, I would like to thank all people, who were participaiting in project.Finally, an honorable mention goes to my familie and friends.Without help of the particular who mentioned above, I would face many difficulties while doing this research.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

V Praze dne 01. 01. 2013 .....

Abstract

This thesis deals with the problem of under-explored space of mid-air text input techniques, that can be used by a standing, mobile user.During this research different motion capture devices and techniques were analyzed, and it was decided to use Microsoft Kinect as a motion caption device.A lot of different text input method were also analyzed. Base on this research 5 different mid-air text input methods, QWERTY,3Push,Gesture,8penKinect and Circular, were designed, mplemented and tested.The results of testing are very intresting, and they are suggesting that that the QWERTY and 3Push are suitable for adaptation. The other techniques need to be improved to compare with this two keyboards.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The history of human society is always narrowly connected with writing. From rock paintings to papyrus and clay tablets, from paper to nowadays mouse and keyboard. During a great part of our evolution, we have been looking for better ways of writing, storing and representing information. This century is not an exception.

1.1 Motivation

Figure 1.1: Microsoft Kinect.[1]

In this thesis, we will introduce and analyze new techniques of text input via a quite modern device - Microsoft Kinect. Kinect is a motion capture device by Microsoft for the

Xbox 360 video game console and Windows PCs. It is based on RGB camera and depth sensor, that enables user to interact whith PC or game console whith gestures and voice commands[2].This Kinect properties was used to allow user to write using only his own body.

It is true that keyboard and mouse or paper and pen are much more convenient way of writing. It is fast, accurate and user is not so wearying in most cases. But this method requires users to sit stationary and it is not posible or comfortable in some cases. If you have ever tried to play games with kinect, you would notice, that it was not very comfortable to use gamepad every time you need to write your nick name, comment or simply to search on Google, or in case of very large displays, when user not within physical reach of keyboard or the display surface.

1.2 Contribution

In this work the following goals were set and achieved:

- Existing moution capture methods and devices were analyzed and base on this analysis, it was decided to use Microsoft Kinect.
- Several text input methods were analyzed
- Base on the analysis of existing text input method, five different mid-air text input techniques, which are QWERTY, Circular, Gest,8penKinect and 3Push were designed
- Using Microsoft's Kinect SDK, this methods were implemented.
- The text prediction API, that is capable to predict text on word level, letter level and UniGlyph level, were created.
- To compare speed, accuracy and physical stress of these methods, they were tested with real users.
- Obtained data were also analyzed. The result suggests that the QWERTY and 3Push are suitable for adaptation. The other techniques need to be improved to compare with QWERTY and 3Push keyboards.

QWERTY keyboard is a graphic equivalent of real keyboard with the same key layout and word level text prediction. For navigation, person uses his left or right hand and to select letter user has to perform, so called "push" gesture.

In the Circular keyboard, letters of the alphabet are shown in a circular arrangement. A pointer line radiating from the center of the circle indicates the currently highlighted letter.

This method is used letter based on prediction and on idea of Circle Keyboard technique in paper [3].

The Gesture and 3Push keyboards are pretty same. Alphabet, in this methods, was divided in to 3 groups using UniGlyph input method. Character input,in those methods, is based on T-9 input method The only difference is that Gesture keyboard is used gestures for each character, and 3Push is used buttons and "push" gestures.

8penKinect is Kinect version of popular 8pen keyboard for android and some other touch devices. With 8pen, you have a circle with 4 quadrants. User draw loops starting and ending in the central fo keyboard. The letter is defined by the starting quadrant, the loop direction and the ending quadrant.

Chapter 2

Analysis

It is impossible to imagine mid-air text input without motion capture. So in the beginning of next chapter, we will describe a different motion capture techniques, in context of main idea of this study. It is also necessary to discuss a text input techniques.

2.1 Motion capture

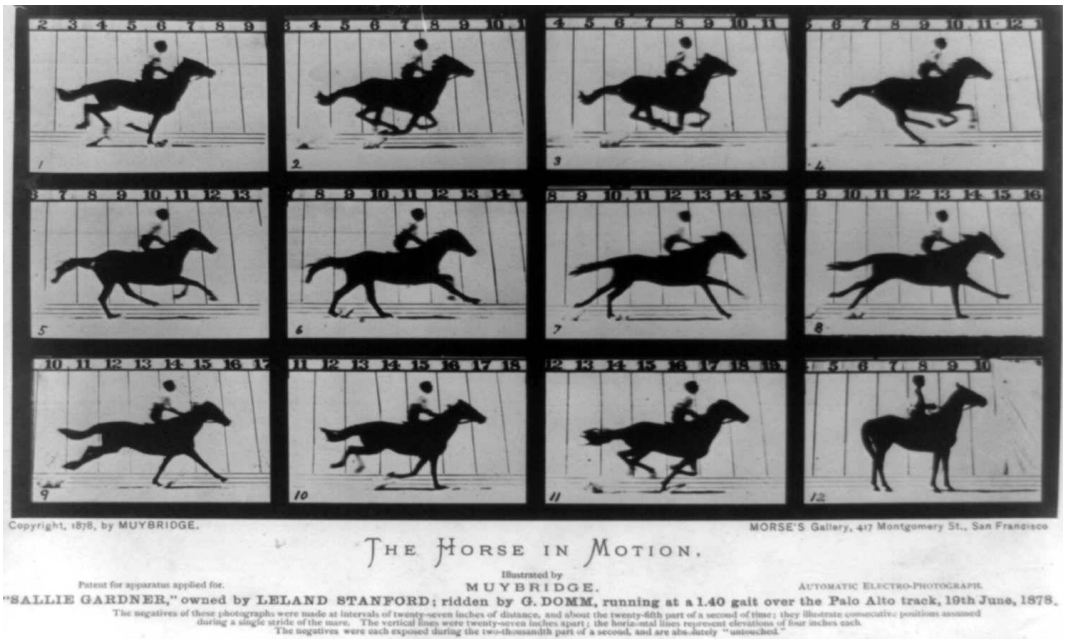


Figure 2.1: The horse in motion.[4]

It is considered that the foundations of motion capture were laid by Eadweard Muybridge in 1872. In that time he was trying to find the answer for the question - doest all four horses

feet are off the ground at the same moment, while horse is running. To find the solution for this problem, Muybridge set an array of several cameras, which took pictures in a different moments. When pictures were played together, it was evident that, in fact, all horses feet are off the ground.

Originally, motion capture technology were developed for military usage in 1970s. But in a mid-1980s mocap became quite popular in entertainment. It has found its niche in the fields of sign language, gesture recognition, rehabilitation, biomechanics, special effects for live action films, and computer animation of all types, as well as in athletic analysis training.

It is worth to mention that, motion capture technology is widely applied in animation. A good example for it is movie "Titanic", where this technologie was used for "filler" characters, or in moments when virtual camera flying over vurtual ship. Many of this scenes would be expensive, difficult or even impossible to create with real cameras, actors and ship.[5] Motion caption gives more realistic movements to the fake characters, than normal computer animation do.

2.1.1 Motion capture Methodology

There are many different systems for motion capture, but all of them can be divided in 5 categories: Acoustical,Mechanical,Magnetic, Optical and Marker-less.

2.1.1.1 Acoustical System

Acoustical systems are base on several sound transmitters, that are placed on actor's body, and receptors that pick up signals from transmitters. Each of the transmitters is activated sequentially, creating a specific set of sound frequencies. The receptors pick up this frequencies and, using the time interval between the emitting of the sound by the transmitter, together with the speed of sound in the particular environment, calculate the distance travelled by the sound. In order to calculate the position of each transmitter in 3D space, a triangulation of the distances between the emitter and each of the receptors is computed.[6]

2.1.1.2 Mechanical System

This system uses set of solid segments, that are attached to the main parts of user's body. Individual segments are connected by joints with potentiometers which measures the orientation. The joint position is determined by the size of the system's segment. Quite often, this system is called exo-skeletal due to way, how the system is attached to the user's body.[8] The main benefits of this system are high accuracy and availability. However, this system limits the users's movements.

Figure 2.2: Mechanical motion capture system.[7]

2.1.1.3 Magnetic systems

[9] Motion caption in this particular system is based on tracking of relative changes in the flow of electric current in a magnetic field. The system consists of several sensors and one transmitter. Transmitter generates a low-frequency magnetic field, that is detected by sensors. Sensors are conected to the electronic control unit, that filters, amplifies and sends detected data, to the main computer. Magnetic systems aren't very expensive, and provides quite high precision, with a sampling rate of 100 frames per second.[11]. The main disadvantage are the limited working space,because of the huge number of cables, and the fact that the magnetic field of this system can be affected by the field of another device, for example mobile phone.

2.1.1.4 Optical Systems

The optical system are the most popular in nowadays, and can be dividing in to two main groups - passive (Reflexive) and active (Pulse-LED). The passive systems are based on cameras and reflexive markers, attached to the tracked person. Cameras are equipped with the set of LEDs, that emits light in the viewing direction of the cameras. For this purpose, infrared light emitting LEDs are used. Light reflects from the markers back to cameras, where light goes through the optical filter, which passes only light with the same characteristic as emmited light. Because of that, cameras is, indeed, sees only the markers. Active optical system are similar to passive one, the main difference is that cameras does

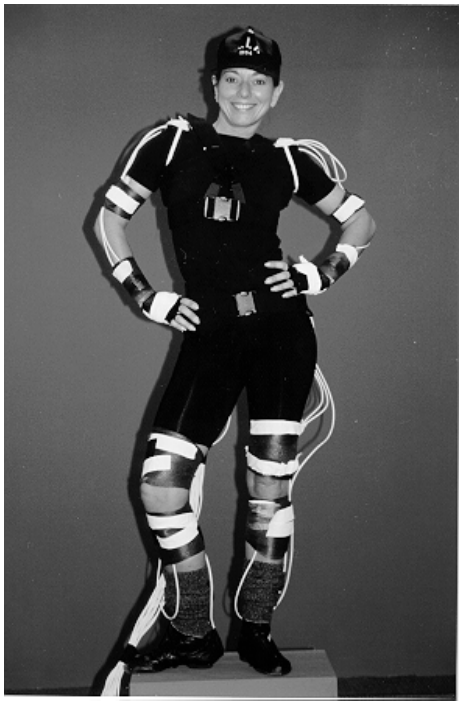


Figure 2.3: Magnetic motion capture system.[10]

Figure 2.4: Optical motion capture system.[12]

not have LEDs, instead markers themselves emit infrared light.

The advantages of this particular systems are very high sampling rate, which allows

to track even fast movements, and freedom of movements, because this system does not requires huge number of cables. However optical system are quite sensitive to ambient light and foreign reflecting objects.[11][8]

2.1.1.5 Marker-less Motion Capture

Unlike other motion capture systems, marker-less motion capture does not requare markers or special suit. These types of systems uses multiple cameras or a camera with a depth sensor to capture the silhouette of a moving person.Than sophisticated computer vision algorithms analyse these silhouette to identify human forms, decomposing these into single, isolated parts used for tracking.[11]. Basically, the whole process of motion caption in marker-less systems relies on software, thus, removing all physical limitation. The main problem of such systems is ambiguity in silhouette recognition 2.5, where software unable to recognize exact human posture.

A good example of such systems is Microsoft's Kinect, that was introduced as low-cost motion capture for the masses. Due to the aforementioned properties of marker-less motion capture and Kinect in particular, it was decided to use Kinect as a motion caption device for this thesis. Kinect will be discussed more in-depth later in 2.1.2.3.

Figure 2.5: ambiguity in silhouette recognition.[8]

2.1.2 Devices for motion capture

There are many different system and devices for motion capture, but most of them are very expensive, cumbersome and requires a special training to appropriately use of them. So they are not suitable for common user and for the purpose of this study, however there are a few that available for public usage, and this chapter will describe them.

2.1.2.1 The Leap

The Leap is a small device that creates approximately 61 cm^3[14] of a 3D interaction space. The Leap is capable of tracking user's hands, fingers and even pen or pencil with

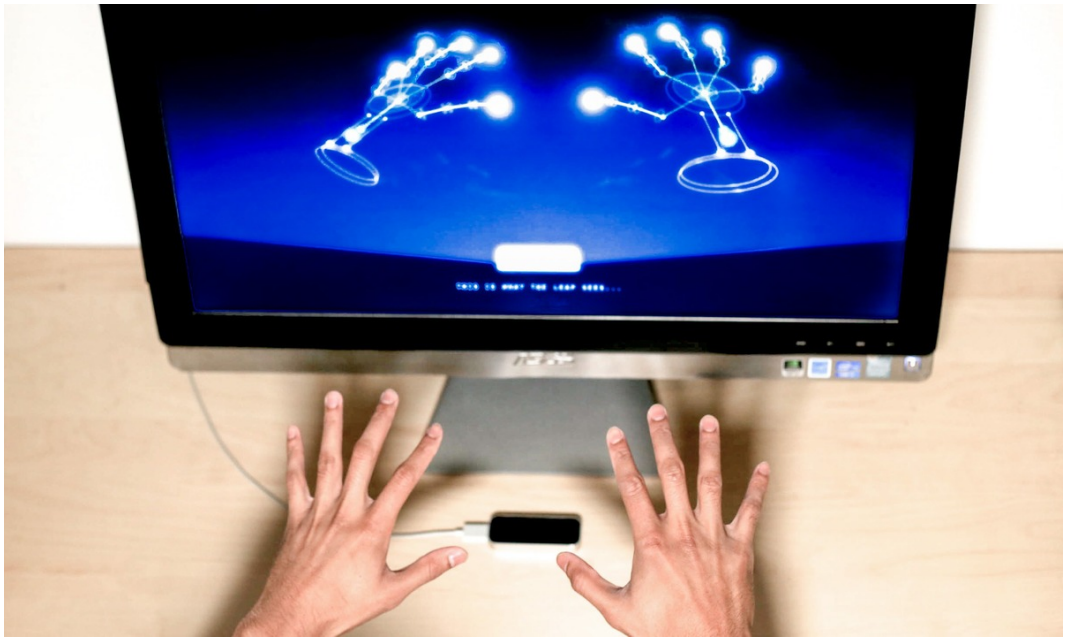


Figure 2.6: The Leap.[13]

very high precision, in fact The leap is 200x more sensitive than similar touch-free devices and technologies[14]. This device gives to user true natural experience, while he is working with different software.

Due to patent pending, very little is known about how actualy the Leap works, and what kind of mathematical algorithms are lies behind the process of tracking user's movements. However, it is known that The Leap uses two CCD cameras and three infrared LEDs to capture depth information. And it is capable to stream 290 frames per second. Device connects to the computer via USB, and does not require any external power source.

2.1.2.2 Wii Remote

The Wii controller looks more like a normal TV remote control, than a classical gamepad or joystick. The main feature of the Wii remote is the ability to track its position in 3D space. Inside the controller is an solid-state accelerometer which allows it to sense:

- Tilting and rotation up and down
- Tilting and rotation left and right
- Rotation along the main axis (as with a screwdriver)
- Acceleration up and down

Figure 2.7: Nintendo Wii.[15]

- Acceleration left and right
- Acceleration toward the screen and away

Wii remote works together with PixArt optical sensor[16], that allows it to determine where the controller is pointing. Using this two key features Nintendo was able to create accurate and natural user interface, which is very popular in home entertainment even today.

The Wii's Sensor Bar is approximately 20cm long and contains ten infrared LEDs, five at each end of the bar. All ten LEDs are divided in to three groups, the first one is pointing slightly inwards, the second one is pointing slightly outwards and the last one in pointing straight. The Wii remote's image sensor sense this light as two bright dots, that have a distance 'x' on a image sensor. The second constant distance 'y' is a distance between the two clusters of light on Sensor Bar. Using this two distances 'x', 'y' and triangulation, Wii is able to calculate the distance between Wii remote and Sensor Bar. The main disadvantage of this system is a limited viewing angle of the Wii Remote.[17] [18]

2.1.2.3 Microsoft Kinect

Kinect was launched on November 4, 2010 and claimed the Guinness World Record of being the "fastest selling consumer electronics device" after selling a total of 8 million units in its first 60 days. [20] Basically immediately after release, community realized that Kinect

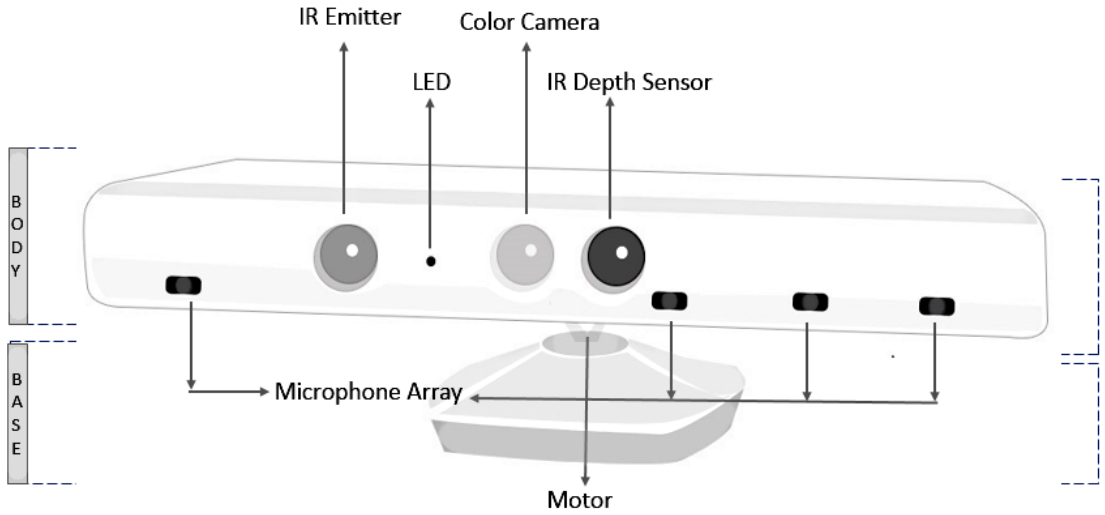


Figure 2.8: Nintendo Wii.[19]

had huge potential that lies far beyond just gaming. Therefore Adafruit Industries decided to announce a competition with price 3000$ for those who would write first open source driver for microsoft kinect. After few days hacker with nickname AlexP created first driver for PC and win the price[8]. Microsoft did not lodge a complaint against hacker, but instead on June 16, 2011 Microsoft released Kinect software development kit for Windows 7. It was meant that SDK allow developers to write Kinecting apps in C++/CLI, C#, or Visual Basic .NET. [21]

How Kinect works.

The Kinect sensor includes the following key components:

- Color camera
- Infrared (IR) emitter
- IR depth sensor
- Tilt motor
- Microphone array

Kinect's color camera is capable of capturing and streaming data in 8-bit VGA resolution (640x480 pixels) with frequency 30Hz or in resolution of 1280x960 pixels with frequency

12Hz. the value of frames per second can vary depending on the resolution used for image frame. The camera has a field of view 63 x 50 degrees, focal length of camera is 2.9 mm. The camera, thanks to it's, properties provides a color image of medium quality. It's main function is to detect red,blue and green color from the source.

The heart of the Kinect is a Depth Sensor, that consist of IR emmiter and IR depth sensor. This two components works together to create depth image. The IR emmiter project infrared light in a "pseudo-random-dot" pattern on area in front of it. The dots are invisible but IR depth sensor is able to see them, and covert this information to depth image by measureing the distance between the sensor and emmited dots. The depth sensor can stream data in resolution 640 x 480 pixels, 320 x 240 pixels, and 80 x 60 pixels with frequency 30 Hz. The depth sensor has a field of view 57 x 47 degrees, focal length of camera is 6.1 mm.

The microphone array contains four different microphones, that are placed in a linear order at the bottom of the Kinect. Three of them are placed on the right side of the dice, and one on a left side. The microphone array is not only capable of picking up surrounding sound, but also locate the source of the audio wave. The microphone array allows to capture and recognize the voice more effectively with enchanced noise suppresion, echo cancellation, and beam-froming technology. It means that Kinect can be a highly bidirectional microphone, that can recognize voice regardless of extraneous noise.[19]

How depth data processing works.

As was mentioned before, the main feature of Kinect is it ability to capture and stream 3D view of the area in front of it, thanks to, infrared emitter and infrared depth sensor. This technology was created by PrimeSense, and the following diagram 2.9 shows how it works [19]:

1. Primesense chip sends command to IR emitter to start project infrared light
2. Primesense chip sends command to IR depth sensor to start capture depth data
3. IR emitter starts sending an infrared light to the objects in front of it
4. IR depth sensor starts to collect data, from the individual light points of reglection.
5. IR depth sensor passes data to the PrimeSense chip
6. PrimeSense chip analyzes data, creates depth image for each frame, and passes it to output stream

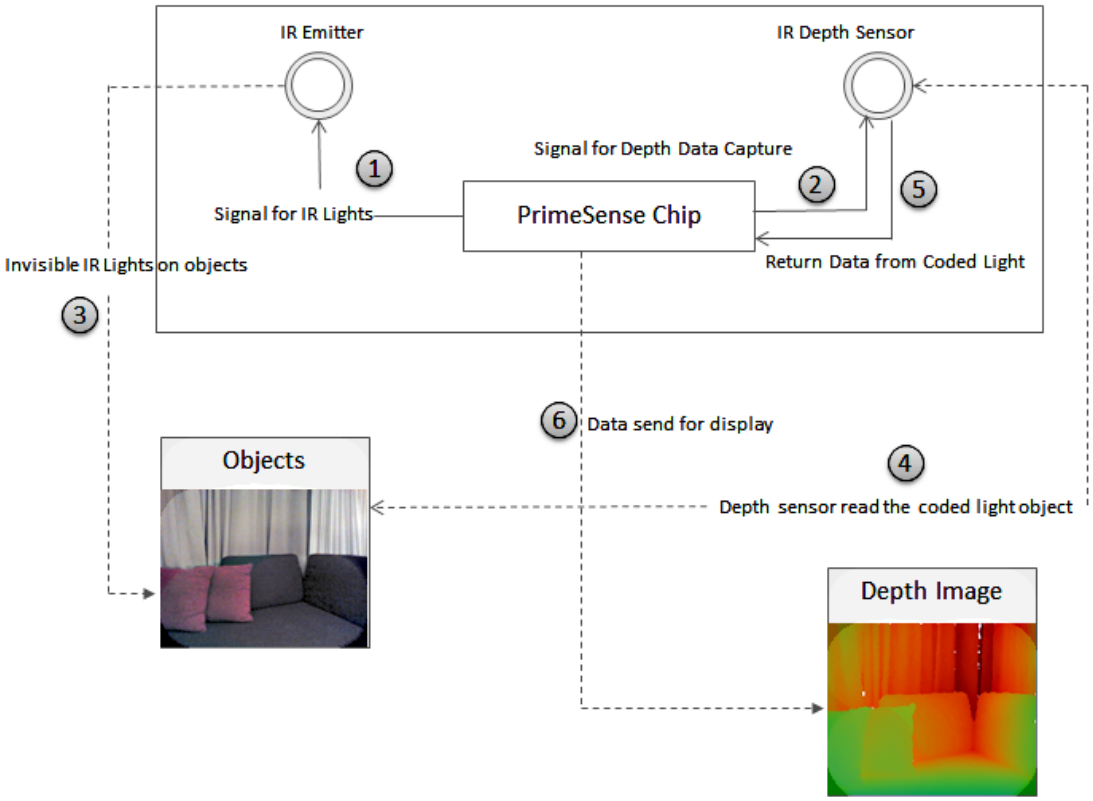


Figure 2.9: Kinect depth data processing.[19]

Software behind kinect.

Kinect uses a machine-learning algorithm by Jamie Shotton, a reseacher at Microsoft Reseach Cambridge in Enggland [22]. Like any other machine-learning algorithm, this algorithm requires training data. For this purpose Kinect developer have collected a huge amount of data from motion-capture in real-life scenarios. Thanks to this, Kinect is able to recognize skeletal joints and distances between them. Kinect can also identify the hidden parts of users body, as long as part of users body is visible for Kinect. It allows Kinect to "see" user behind a furniture in the room or behind another user.

2.1.3 Summary

All aforecited devices were considered for the role of moution capture devices for this thesis, but it was decided to use Microsoft Kinect. The Leap is a quite an intresting device and it is even more precise than Kinect. It has good documentation, SDK and support

from developers and community, but it is designed to use near the computer and not by a standing, mobile user, furthermore The Leap not yet available in Czech Republic.

The Wii Remote is also an astonishing device, but it is able to track only the Wii remote itself, not the user, so he have to hold the device in order to interact with computer. This fact strongly limits user's movements, which Wii can recognize. Wii Remote does not have any official support, SDK or API, however there are some third part libraries, but they are not so great as official one for Kinect or The Leap.

On the other hand, Kinect is able to recognize and track whole user's body at a distance up to 4 meters and in any light conditions. Kinect has a great support from Microsoft and community. Microsoft have published SDK, that allow developers to write apps for Kinect in C++/CLI, C#, or Visual Basic .NET. This SDK has a great documentation with tutorials and video lessons. There are also a huge number of different APIs from community. Besides, it is possible to buy Kinect, literally, in any big gaming or electronic store, and it is not very expensive, for the device with such great capabilities.

2.2 Text input methods

At the current moment there are a numerous text input method exist. Some of them were created to improve performance, some were created for people with handicaps, and some to meet the requirements of new devices, but all of them belong to one of the five main categories which are [23]:

- keyboards
- text recognition
- unistrokes
- speech recognition
- gesture recognition

Each category has a number of sub-categories of which the most relevant ones are shown in figure 2.10

2.2.1 Virtual keyboards

Virtual keyboards are applications that allow user to input characters. Nowadays, virtual keyboards are very common on smart devices with touch screen, because they can be easily adapted to a different screen size and familiar to every user. Typically it is a grid keyboard

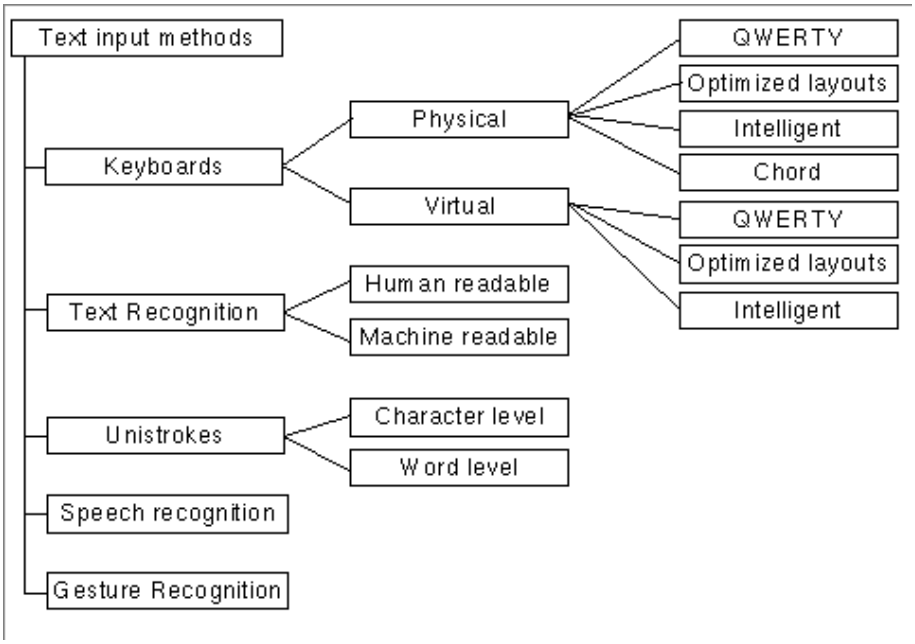


Figure 2.10: Text input methods.[23]

with traditional qwerty layout, but there are many differen alternative keyboards, that allowing to input text differently, for example, using gestures or handwriting. Virtual keyboards, can be operated not only with touch screen, mouse or physical keyboard, but also with special devices for handicapped persons, it might help them to use computer more comfortably. Virtual keyboards may be used in some cases to reduce risk of keystroke logging. For example, Westpac's online banking service uses a virtual keyboard for the password entry, as does TreasureDirect[24].

2.2.1.1 QWERTY

The qwerty-layout was patented in 1869 and to this day it is the most common keyboard layout. Competing arrangements were invented during this time, for example Dvorak Simplified Keyboard and Colemak layout. Some of them were proved to be more effective than qwerty, but qwerty is still the most common arrangement of letters, and the prevalent one on computers keyboards.

The difficulties in learning QWERTY- virtual keyboards may have many explanations. Three very convincing reasons given by Gopher and Raij are[23]:

- The visual appearance of the characters has no connection to the motor activity needed for pressing the key.

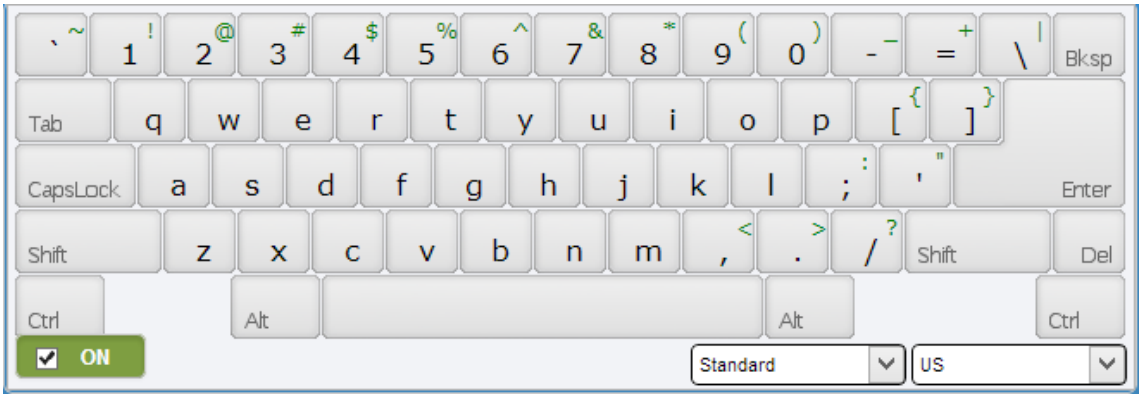


Figure 2.11: QWERTY keyboard.[25]

- The keyboard has no structure that could be easily converted into a cognitive internalization of the keyboard layout and typing activity.
- The number of keys and especially the number of key to key transitions is simply too large to be learned swiftly.

Moreover, qwerty is not optimal for single-pointer input device, there are number of better designs. However, qwerty frequently used as people are quite familiar with it, and it takes a rather long time to learn a new layout.

KeyStrokes R 4.

To improve performance and comfort of virtual keyboards whith QWERTY layout, Origin Instruments had developed KeyStroke R 4, which is available for MAC OS. This virtual keyboard is highly configurable, contains many features that make typing easier, offers multiple keyboard layouts and also allows users to create his own layout using LayoutKitchenTM. This aplication can be control by variety of devices including mouse, trackball, head pointer and touchpad. The clock action is also customizable and can be change to single click, double click, right click, drag and etc.

KeyStrokes R 4 includes an integrated next word prediction, multi-word prediction, learning with automatic spell-checking and a dictionary editor. Aplication has an audio feedback including speaking the typed words.[26]

Grid Keys.

Figure 2.12: KeyStrokes R 4 keyboard. [26]

Figure 2.13: Grid Keys keyboard. [26]

Virtual keyboard Grid Keys was developed for Windows OS, and as well as KeyStrokes R 4, Grid Keys can be controlled by alternative pointing devices, such as trackball, joystick, head pointer and touch screen. This virtual keyboard alloes user to use Windows application without hardware keyboard or mouse. It also has an integrated word completion system and allows user to configure keyboard screens (called grids), what is more, not only the placement of the keys can be changed, but special keys can be added or removed. The application is available in several languages and has a speech output.[26]

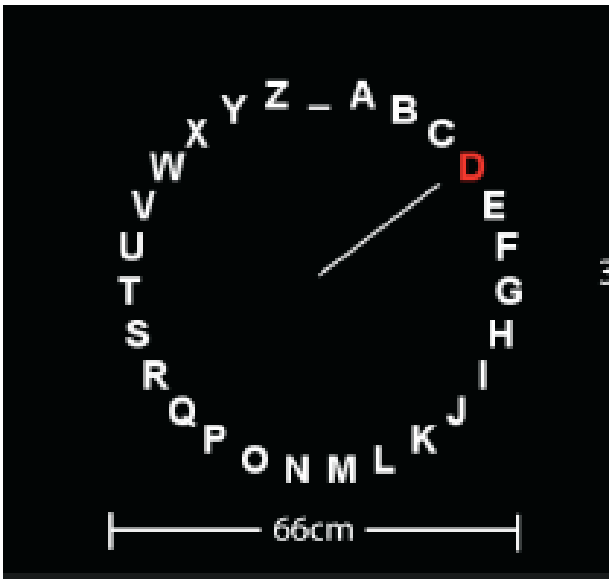


Figure 2.14: Circular keyboard.[3]

2.2.1.2 Circular keyboard

Quite interesting concept of circular keyboard was described in article [3]. This article was engaged in research of various mid-air text input methods. Inspiration for the technique came both from a technique developed for touch wheel text input, and from a similar technique used in the Nintendo Wii Game "Super Monkey Ball."

In the Circle technique, letters of the alphabet are arranged in a circular as shown on 2.14. User uses handheld device to control a line that radiating from the center of the circle and indicates the currently letter. User moves the pointer to highlight the desired character and than presses a button to select that character. [3]

2.2.2 Unistrokes

Latin alphabet has it's own limitation and one way to overcome some of them is to design a new alphabet. It is known that, trained user can write one stroke, even a complex one, much faster than several simpler ones. Using this information Goldberg and Richardson had created the Unistroke alphabet that is, indeed, a forefather of whole family of unistroke text input methods that have appeared lately. Novadays unistroke text input methods have evolved beyong the original one charakter per stroke rule, therefore, can be divided in two groups: character-level unistrokes systems and word-level unistroke systems. [23]

2.2.2.1 UniGlyph

Figure 2.15: UniGlyph character set.[27]

UniGlyph was originally created for people with motor impairments and for the able-bodies users of small devices. This unistroke method was designed to meet the principles of "learn once, write everywhere". To choose primitive representing each letter, developers set two main criteria [27]:

- Effective prediction: UniGlyph is an ambiguous text input method, the set of primitives must be chosen in a way to optimize the result of the disambiguation system.
- Easy memorizing: the choice of the primitive representing each letter should be logical to make learning easy.

To find the solution, that satisfies these requirements, several sets of primitives were created based on statistical study. The two best solutions were then tested online by 117 users. The result of this reseach is a character set, that represented on 2.15.

2.2.2.2 8-pen

8pen virtual keyboard for Android mobile devices with touch screen. 8pen is attempting to replace classical keyboard layout on which base on touch screen mobile input. This virtual keyboard introducing quite intresting way of thinking about text input on small displays. Instead of tapping the screen with high acurace to produce a character, 8pen allows user to write like he woud with pen.

To produce a character user have to put his finger on the center of the keyboard, then enter any of 4 sectors, and pass throught either 1, 2, 3 or 4 adjacent sectors in either clockwise or anticlockwise directions, and then again return to the center again. Order of the letters and the side on which whey placed along the edges, determine the number of sectors that user have to pass through, and the direction of the movements.

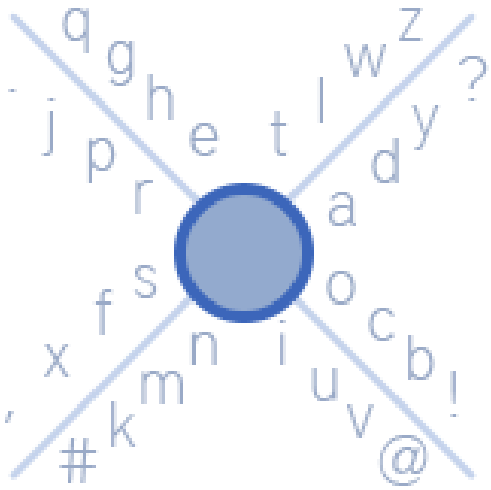


Figure 2.16: 8-pen keyboard.[28]

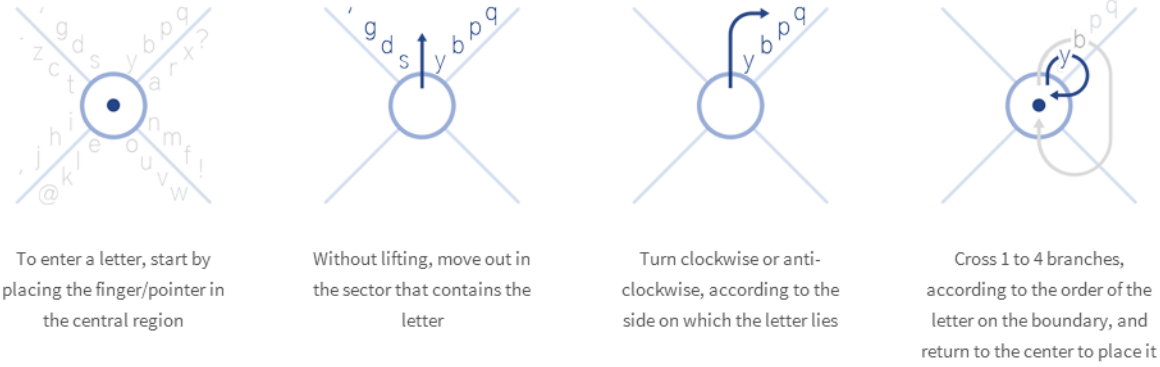


Figure 2.17: 8-pen character input.[28]

2.2.2.3 T9

T-9 stands for Text on 9 keys and it is a predictive text technology for mobile phone, that was developed by Tegic Communication. T-9 is is still quite common on novadays mobile phones with touchscreen, but originally it was developed for the mobile phones that contains 3x4 numeric keypad. T9 was used on variety of phones including Verizon, NEC, Nokia, Samsung Electronics, Siemens and etc. It was also used by Texas Instruments PDA

Avigo during the late 1990s. The main competitors of T-9 is iTap created by Motorola.

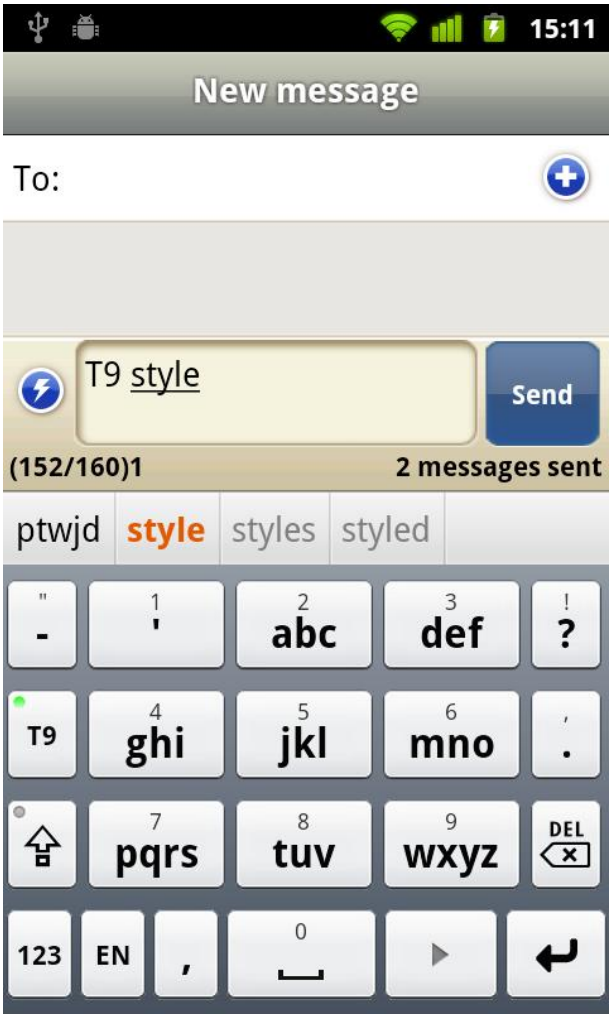


Figure 2.18: T9.[29]

The main T9's objective is to make typing messages on mobile phone easier. It uses groups of letters on each phone key in combination with a dictionary of words, allowing words to be entered by single keypress for each letter, instead of multi-tap approach. T9's searches words in dictionary by the sequence of keypresses, and then order founded words by frequency of use. This text input method speeds up the process of writing by getting familiar with the words and phrases that user commonly use, and incresing it's raiting in the list of founded words. T9's dictionary can be manualy expanded by adding missing words via multi-tap.

Lets say users want to type word "the". To do so, he have to press 8 then 4 and then 3, and the display would display 't' then 'th' then 'the'. But if users want to write word 'fore',

situation will be slightly different. User would enter 3, 6, 7 and finally 3, predictive algorithm may select 'Ford'. Pressing the key for 'next' (typically the '*' key) might select word 'close', and finally 'fore'. If 'fore' is selected, then the next time user presses the sequence 3-6-7-3, the first word in the list will be 'fore'.[30]

2.2.3 Summary

In the previous chapters several text input methods were analyzed. These methods were the basis for creating QWERTY, Circular, Gest,8penKinect and 3Push mid-air text input methods. Mid-air QWERTY is based on standart QWERTY layout. The idea of word prediction and visualization of predicted words were inspired by KeyStrokes R 4 and Grid Keys virtual keyboards.

Mid-air Circular keyboard was inspired by circular keybord from article [3]. But letter prediction was added to mid-air version of this keyboard. The size of each circle sector depends on letter probability. The higher probability the bigger letter's sector. This modifications should facilitate letters selection.

The character input in Gest and 3Push mid-air virtual keyboards is based on T-9 input method. However Gest and 3Push has only 3 groups of letters instead of 9. Alphabet, in this methods, was divided in to 3 groups using UniGlyph input method. The method of alphabet division, will be discussed in chapter 3.3.

8penKinect is baisicaly a mid-air representation of touch screen 8-pen. But in case of mid-air 8penKinect user can't "detach" his hand from devices display, so a few modifications, that will be described in 3.5, were made.

Chapter 3

Design

In this chapter the design of 5 mid-air text input methods will be discussed which are: QWERTY, Circular, 8penKinect, Gesture and 3Push. It is worth to mention that mid-ait text input is quite modern concept, and there are only few studies that have investigated this problem directly.Despite this we have create a set of requirements for mid-air text input software:

- Movements, that user perform for text input, have to be as less physically exhausting as it possible.
- Movements have to be easy to remember
- Number of movements for text input should be minimal
- The UI of application have to be big enough. to be visible from a distance of 1,5-2 meters
- Mid-air text input software has to provide visual feedback.

3.1 QWERTY

The QWERTY keyboard is a classical virtual keyboard with qwerty layout. To reduce a number of key presses, word completion was added. The QWERTY keyboard has 26 buttons for each letter from english alphabet, plus one for backspace and one for space. This keyboard does not allow to write capital letters, numbers or special symbols.

The keyboard windows is big enough to be visible from distance 1,5-2 meters even on small displays.The size of keyboard is 800x1350 pixels,the size of buttons is 130x130 pixels, with exception of space and backspace, which size is 150x150 pixels.Font size is 36 pixels.For

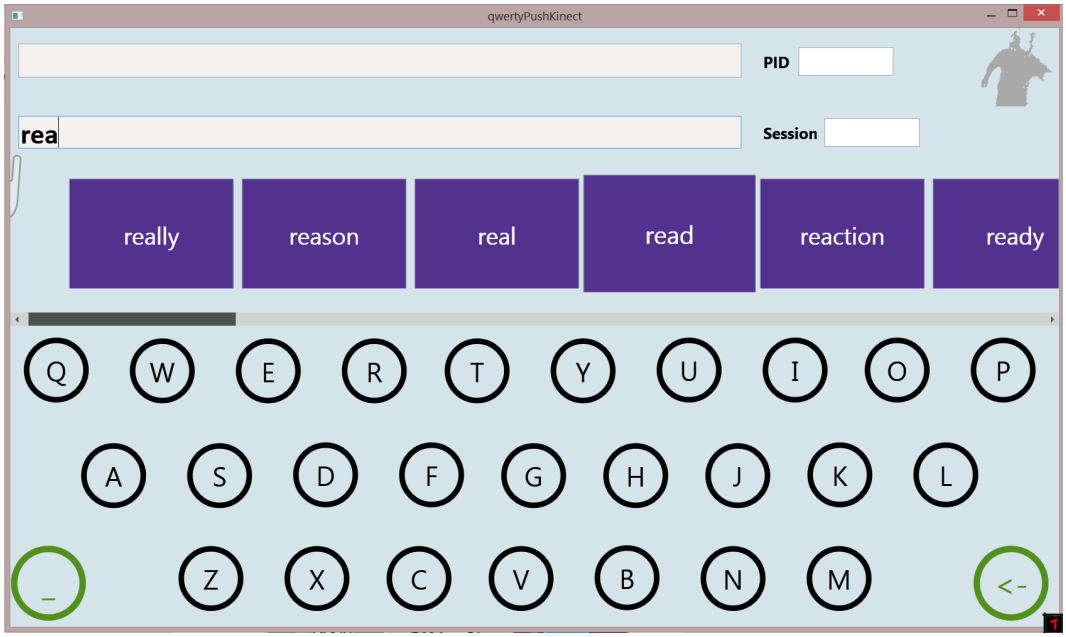


Figure 3.1: Mid-air QWERTY design.

navigation user controls cursor with his left or right hand, to select button user perform so called “push” gesture 4.1. To scroll list of predicted words user has to perform “grip” gesture 4.1 and move hand to right to scroll to he right or move hand to the left to scroll to the left. To select word from list user also has to perform “push” gesture.

Figure 3.2: Push gesture.[31]

Some times, while user perform “push” gesture the position of the hand can slightly change, and wrong button can be selected. To avoid this problem each button has,so called,

Figure 3.3: Grip gesture gesture.[31]

"sticky" effect. It means, that while doing push gesture, the cursor sticks to the nearest button.The application has few types of visual feedback. The first one is button animation - each button changes it's color and size, while being pressed. the second one is cursor animation : cursor changes it's shape on "grip" gesture and provide animation on "push" gesture 3.4.The last one is depth stream in right upper corner of application 3.1.The overall process of text input with mid-air qwerty can be described in next steps:

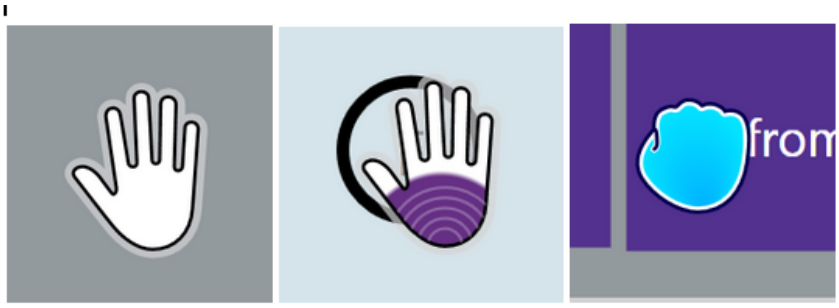


Figure 3.4: Kinect cursor animation.

1. User places cursor over desired letter
2. User perform "push" gesture to input letter
3. Now user has 2 options
   (a) Go to step 1 and continue with next letter

(b) User scroll list of predicted words with “grip” gesture and then select word with “grip” gesture

3.2 Circle keyboard

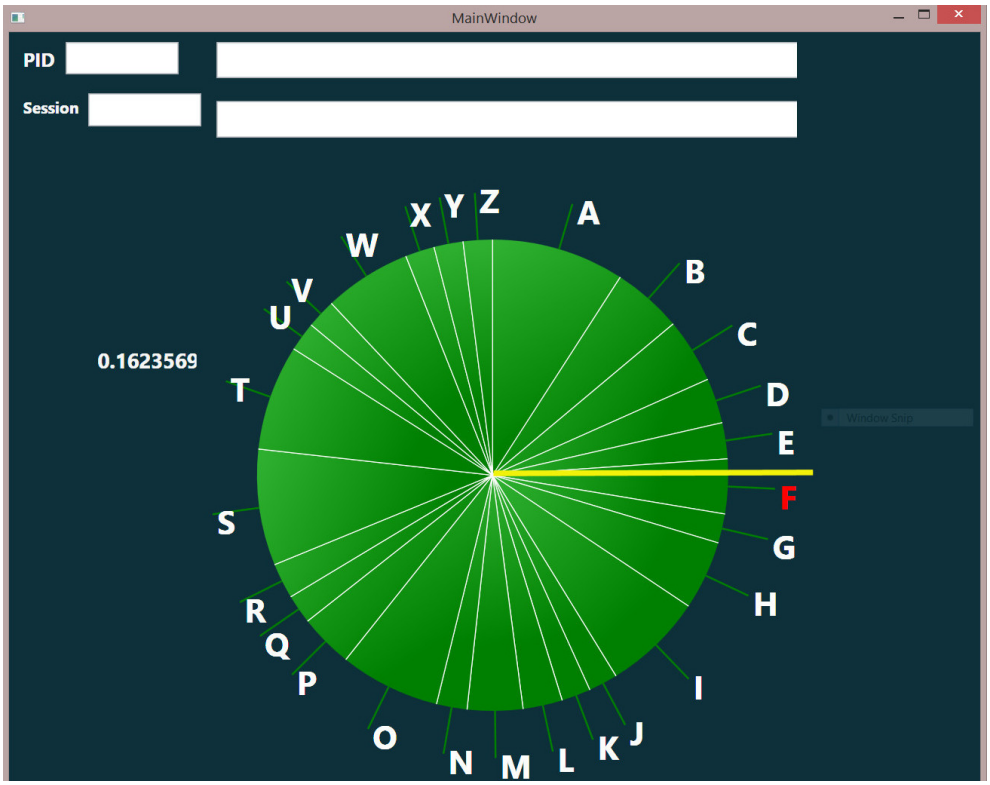


Figure 3.5: Mid-air Circular keyboard design.

The Circle keyboard is based on circular keyboard from article [3]. Letters of the alphabet are also shown in a circular arrangement. A pointer line radiating from the center of the circle indicates the currently highlighted letter. But we have added letter prediction to this input method. The size of each circle sector depends on letter probability. The higher probability the bigger letter’s sector.

The size of the application main windows is 860x1000 pixels, the size of circular keyboard is 500x500 pixels, font size is 22 pixels for text and 35 pixels for each letter.So the application is big enough to be visible from distance.To select letter, user rotates pointer line with his right hand. to input letter user have to place pointer line over letter’s sector and do LeftHello gesture. To input space user have to perform RightHello gesture. To delete last letter user have to perform HandsTogether gesture.

Each time a user enters a letter, the size of each sector changes .This should not only helps user to write text, but also works as good visual feedback. If user did not write anything yet, or place a space, the sizes of sectors return to it's default value. the default values are based on relative frequencies of letters in english language.

3.3 Gesture keyboard

The Gesture keyboard is quite a unique keyboard, that was designed, during this study. Gesture keyboard is based on ideas of T9 and UniGlyph. In this input method, the whole set of letters in english alphabet was divided into 3 subsets,exactly as in UniGlyph, which are:

1. E F J L U T I H
2. B R C O P S Q G D
3. A K M V W Z Y X

Frankly speaking, letters belong to the first subset, if they have only vertical or horizontal elements, to the second subset, if they have roud element and, finally, to the third subset, if letters have oblique element. As was mentioned before, studies indicate that, UniGlyph method is fairly comfortable and easy to remember for users. In gesture keyboard each group of letters has it's own gesture. To input a letter from first group user have to perform RightHello gesture, to input letter from second group, user have to perform HandsTogether gesture, and to input letter from the third group, user have to perform LeftHello gesture.

Because Gesture keyboard works in the same manner as T9, user does not actually see letters, while he input text, but only symbols that represent this particular subset of letters. In T9 input method each group of letters are represented by number from interval [2,9].In gesture keyboard the first group is represented by symbol "|", the second by letter "O", and the third by symbol "/". After user inputs pattern, that consists only from symbols that represents each subset, he has to select actual word from word list, with his right hand and "push" gesture.

The description of gesture keyboard can sounds complicated, so let's make an example. Suppose user wants to input word "Hello". To do that, uset have to perform RightHello gesture four times, and then perform HandTogether gesture once. After that, user will see pattern "| | | | O" and list of word that can match this pattern. To actually input word, user have to navigate cursor, with his right hand, to the word "hello" in wordlist, and perform "push" gesture.

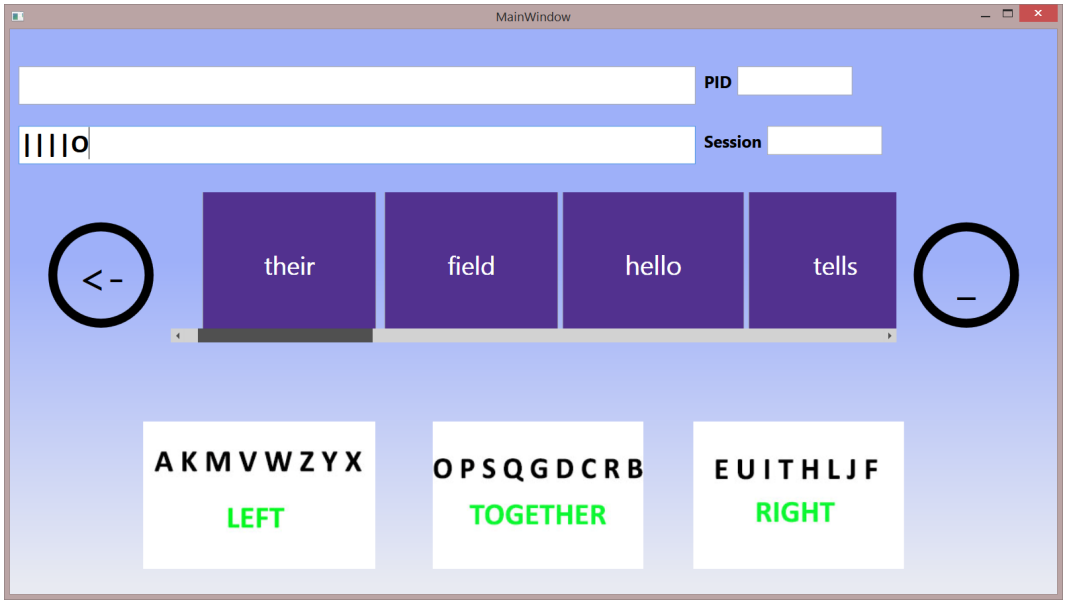


Figure 3.6: Mid-air gesture keyboard.

The size of application windows is 1280x720 pixels, fontsize is 33 pixels.So the application is big enough to be visible from distance. Changes in wordlist works in combination with cursor and word buttons animation works as pretty good visual. As additional visual feedback there is a depth image of user in upper left corner. feedback.The button "<-" and button "_", works as backspace and space respectively.

3.4 3Push keyboard

The design of 3Push keyboard is very similar to the design of Gesture keyboard. 3Push is also based on ideas of UniGlyph and T9 and works in the same way as Gesture keyboard does, but with one small but noticeable exception. Instead of assigning each group of letters to the gesture, 3Push keyabord has 3 buttons, one for each subset of letters. So, to input word user has to use only buttons.

To press the button user has to do "push" gesture, while holding cursor over appropriate button. 3Push keyboard also has 2 special buttons- "_" for space and "<-" for backspace.The size of application window, font size and visual feedback is the same as in Gesture Keyboard.

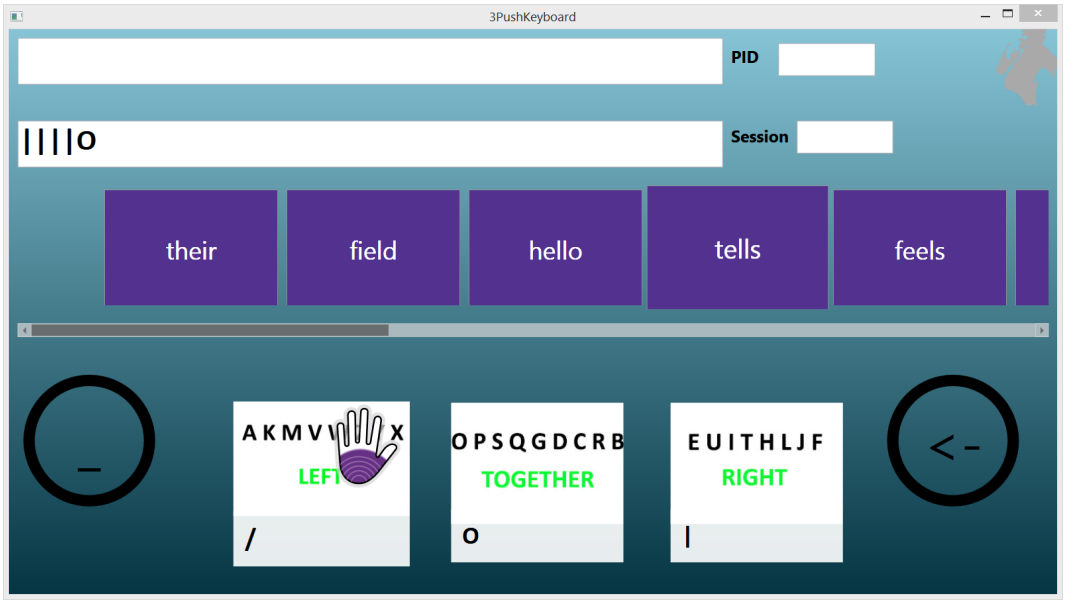


Figure 3.7: Mid-air 3Push keyboard.

3.5 8penKinect

Our 8penKinect keyboard is based on popular virtual keyboard for android devices - 8pen.To avoid confusion, we will use 8penKinect when referring to the our design and 8pen reffering to original keybord for android.

The size of application main windows is 700x500 pixels,font size is 30 pixels. Cursor size is 50x50 pixels. The overall application size is smaller, then sizes of qwerty and circular keyboards, but still it's big enough to be visible from distance 1,5 meters. To write user controls cursor with his right hand and enters letter in the same way as with 8pen. A character is produced by placing cursor in the center, entering any of the 4 sectors, and then passing through either 1, 2, 3 or 4 adjacent sectors in either clockwise or anticlockwise direction, before returning to the center. The order of the letters along the edges, and the side on which they are placed, indicate the number of sectors to be passed through, and the direction of the movement, respectively.

8pen was originally designed for devices with touch displays, but in case of mid-air text input user can't "detach" his hand from devices display. To solve this problem, several modifications were made:

1. At the beginning of text input, user have to place cursor in the center.
2. To delete the last letter, user have to move cursor from center to left and then return back to the center.

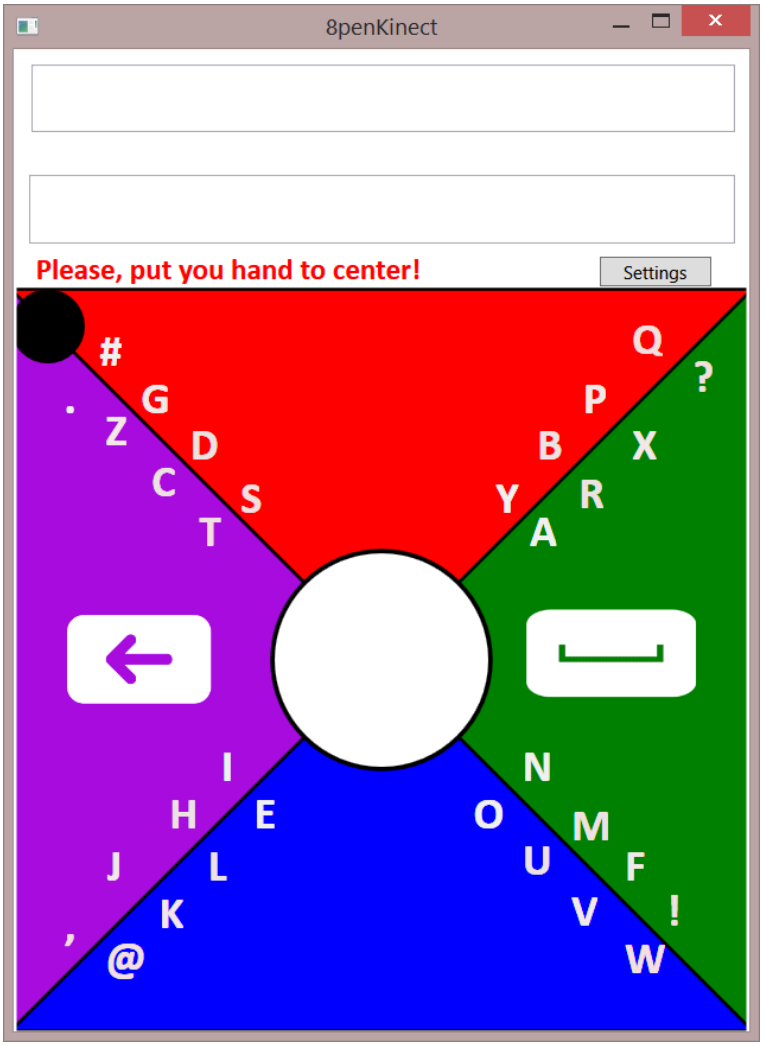


Figure 3.8: Mid-air 8penKinect keyboard.

3. To enter space, user have to move cursor from center to the right and then return back to the center.

The application has two types of visual feedback. The first one is warning messages, than inform user about incorrect strokes, and that he has to place cursor in the center at the beginning of text input.The second one is sectors discoloration - each sector changes it's color when user moves cursor through it.

Chapter 4

Implementation

As was mentioned before, for our mid-air text input methods we decide to use Microsoft Kinect. Besides advantages of Kinect, that was described in chapter 2.1.2.3, we decide to use it mainly because Kinect is publicly available, has a good community support and because Microsoft Kinect SDK allow to write programs for Kinect in C# and WPF.

One of the most problematic part of implementation was gesture recognition. The Microsoft Kinect SDK is quite young, and there are still only small open source projects that allows gesture recognition. One of them is Kinect Toolbox and we will discuss it in the next chapter.

The second challenging part was word and letter prediction. There are plenty of different software and API for word and letter prediction but most of them are not for free or too complicated, that's why we decide to implement simple but efficient TextPredictionAPI, that can predict words on letter and word level and it also able to predict words by UniGlyph pattern.

There are also a few interesting implementation notes in each of implemented input method.

4.1 Microsoft Kinect SDK

As was mentioned before, Kinect was originally created for Xbox 360 as a new type of controller for gaming. But in June 2011 Microsoft have released the beta SDK for non-comercial use, and then in February 2012 fully supported version 1.0 for commercial and business us was published. This SDK is designed for Windows applications, and it means that Microsoft has opened Kinect functionality to much wider variety of uses, for example, Kinect can be used in education, healthcare and transpartation. The requirements for developing with Microsoft Kinect SDK are:

- A 32-bit or 64-bit dual-core processor that's 2.66 GHz or faster
- 2 GB of RAM
- A dedicated USB 2.0 bus
- Microsoft Windows 7, Windows Embedded Standard 7 or Windows 8 Developer Preview
- Microsoft Visual Studio 2010 (Express or other versions)
- .NET Framework 4.0
- Microsoft DirectX SDK (June 2010 or later version)
- Runtime for Microsoft DirectX 9
- A Kinect for Windows sensor with special USB data/power cabling

It's worth to mention, that Microsoft Kinect SDK not only gives ability to work with Kinect, but also offers a few UI WPF controls, wich are: [32]:

- **Kinect User Viewer** is able to visualize the depth pixels of identified user or users. KinectUserViewer can serve as indicator, that Kinect actualy recognize users as a human and is able to separate him from surroundings. If Kinect has not identify user as a person yet, KinectUserViewer will show nothing.
- The **KinectTileButton** is one of the basic, but usefull, controls available. It looks like Windows 8 tile, with label and a content that is just like the content of the regular Button control. When user moves cursor over KinectButton it changes it's size to indicate the "hover" state. The cursor also changes it's size. If users starts to performe "push" gesture, the cursor will begin to fill up with purple pattern 4.1. When users presses the KinectTileButton, it will play basic animation, that serve as visual feedback.
- The **Kinect Circle Button** works a lot like the KinectTileButton. The KinectCircleButton looks like RoundButton of Windows Phone. When KinectCircleButton is in "hover" state, it also changes the size of hand cursor and button itself.

4.2 Kinect Toolbox

Kinect Toolbox is a set of useful tools for developing with Kinect for Windows SDK (1.7).Kinect toolbox has such features as [33]:

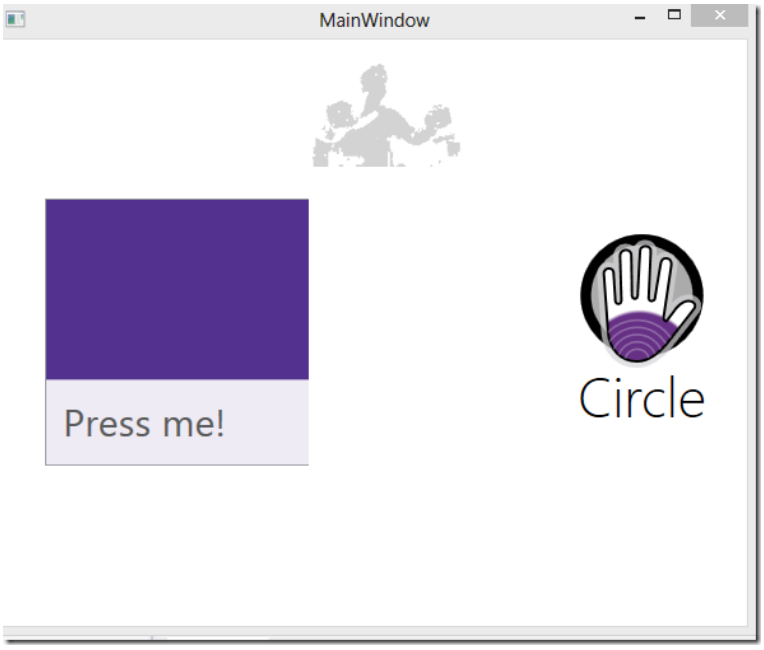


Figure 4.1: Microsoft Kinect SDK UI elements.[32]

- *SwipeGestureDetector* that can detect the following gestures
  - SwipeToLeft
  - SwipeToRight
- *AlgorithmicPostureDetector* can detect the following postures:
  - None
  - HandsJoined
  - Left Hand Over Head
  - Right Hand Over Head
  - Left Hello
  - Right Hello
- *BarycenterHelper* is able to detect if skeleton is static or is moving.
- *SkeletonDisplayManager* allows to draw a skeleton frame on top of a WPF canvas.
- *Voice Commander* can and raise an event when it detect one of the predefined words (using the microphone array of the sensor).
- *ContextTracker*: detects if user is stable.

- *EyeTracker:* allows to detect if the user is currently looking at the sensor.
- *MouseController*:This controll allows to controll cursor with kinect.
- *MouseImpostor*: Controller that replaces mouse cursor.

4.3 TextPrediction API

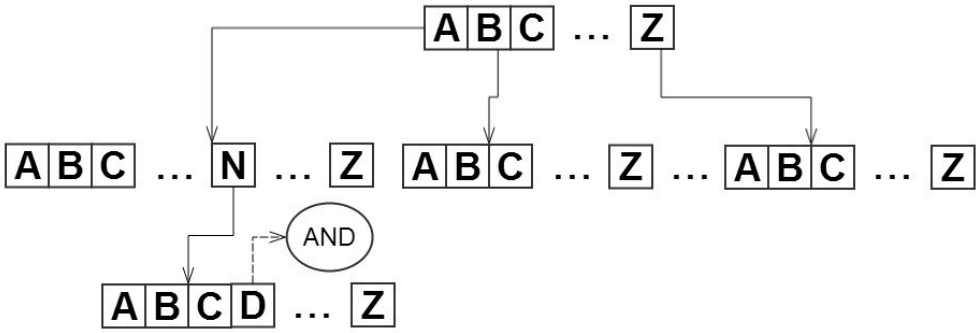


Figure 4.2: Text Prediction API data structure.

For the sake of our research, the simple and quite efficient algorithm for word completion and letter level prediction was created. The algorithm uses publicly available dictionary of english words and it has almost 9000 english words. Each of this word in the dictionary has a number that identify it's frequency . Each line in the dictionary has the following format: [**<word> <word_frequency>**].

At the beginning of work, TextPrediction API reads each line from dictionary, and creates a searching Tree structure, that hold whole dictionary together. Each node (we will call them "shelves") in the Tree is a Map data structure.Each key in the Map is a letter, and each value is an object (we will call such objects "nodes") that hold reference to next shelf, current letter and information about letter frequence. The letter frequency is a sum of words frequencies,that were traversing through this node.If it's possible to create word from letters that lies on the path from one of the root nodes to the current node, that this node also holds this particular word and information about word frequency.We will call such nodes - wordNodes.

The algorithm for adding word to the tree is quite straightforward: for each letter in the word, algorithm traversing tree to a proper shelf, if at current shelf does not have node

with such letter, algorithm create new node, otherwise algorithm will continue traversing tree. At the end algorithm creates wordNode, with current word.

```
public  void addWord(string word,int wordFrequency)
 {
    word = word.ToLower();
    Dictionary<string, Node> nextShelf = rootshelf;
    int iter = 0;
    Node tmp,create;
    foreach (char c in word)
    {
      iter++;
      if (nextShelf.ContainsKey(c.ToString()))
      {
         tmp = nextShelf[c.ToString()];
         tmp.letterFrequency += wordFrequency;
         nextShelf = tmp.nextShelf;
       }
        else
       {
         if (iter == word.Length)
         {
           create = NodeCreator.createNode(c.ToString(), word, wordFrequency);
         }else{
           create = NodeCreator.createNode(c.ToString(),wordFrequency);
         }
      nextShelf[c.ToString()] = create;
      nextShelf = create.nextShelf;
     }
   }
}
```

The function for word completion, is quite simple too. As input parameter it gets part of the word and traversing tree to a proper shelf. After that algorithm runs standard BFS that will return list of possible words. At the end algorithm sorts words by it word frequencies.

```
  public List<WordNode> wordCompletion (string word)
      {
          if (word.Length < lattersToStart) return null;
```

```
        List<WordNode> result = new List<WordNode>();

        Dictionary<string, Node> currentShelf = getShelfLevel(word);
        if (currentShelf == null) return null;
        Queue<Node> queue = new Queue<Node>();
        foreach (Node n in currentShelf.Values)
        {
            queue.Enqueue(n);
        }
        result = BFS(queue);

        result.Sort();
        return result;
    }
```

The algorithm for letter level prediction is a bit more complicated. As well as function for word completion, this function also gets as parameter part of the word and traversing tree to a proper node. The shelf on which point this node, contain all possible letters. The probability of each letter quals to 4.1, where *frequency_summ* is sum of all letter frequencies in this shelf. But there is a little catch. In this research we are using this function for determin sizes of sectors in Circular keyboard. And there is two issues with that.The first one, is that the searching tree can not guarantee that all letters from english alphabet will be at the current shelf. In this case, we are adding missing letters from special list, that contains all english letters and it's probability. The probability of each letter in this list is based on relative frequencies of letters in the English language [34]

$$letter\_probability = letter\_frequence/frequency\_summ * 100 \tag{4.1}$$

The second one, is that the probability of some letters can be too small and hence the size of sectors in Circular keyboard can be too small, and user will be unable to use it. To solve this problem we decide to set the minimum probability of each letter to 2%. If the probability of letter is lower than 2%, the algorithm founds *delta* 4.2. Then algorithm found letter with maximum probability, subtracts delta from it's probability, and add delta to the probability of the current letter with probability lower that 2%.

$$delta = 2 - letter\_probability \tag{4.2}$$

The algorithm for word completion base on UniGlyph pattern is basically the same as algorithm for word completion, but with one exception: because each UniGlyph character,

represent several letters at once, the algorithm traversing the searching tree by several paths at once, and then runs BFS from each end node.

```
public List<WordNode> predictByPattern(string pattern)
{
  string currentPatternLetter;
  string[] currentpattern;
  List<Dictionary<string, Node>> shelflist =
          new List<Dictionary<string,Node>>();
  List<Dictionary<string, Node>> nextshelflist=null;
  List<WordNode> resultnode = new List<WordNode>();
  shelflist.Add(rootshelf);
  int iterator = 0;
  foreach(char s in pattern)
  {
    iterator++;
    currentPatternLetter = s.ToString();
    if (patterns.ContainsKey(currentPatternLetter))
    {
      currentpattern = patterns[currentPatternLetter];
      nextshelflist = new List<Dictionary<string, Node>>();
      foreach (Dictionary<string, Node> iteratedshelf in shelflist)
      {
         foreach (string lt in currentpattern)
           {
             if (iteratedshelf.ContainsKey(lt))
              {
               if (iterator < pattern.Length)
                 {
                   nextshelflist.Add(iteratedshelf[lt].nextShelf);
                 }
                  else
                 {
                   if (iteratedshelf[lt].GetType() == typeof(WordNode))
                   {
                     resultnode.Add((WordNode)iteratedshelf[lt]);
                    }
                  }
                }
```

```
                }
              }
            shelflist = nextshelflist;
          }
          else
          {
            return null;
          }
      }
      resultnode.Sort();
      return resultnode;
  }
```

4.4 Implementation notes for text input methods.

All text input methods were implemented using Microsoft Kinect SDK. The QWERTY and 3Push keyboards were created using standard elements from Microsoft Kinect SDK, wich are: KinectUserViewer, KinectCircleButton, KinectRegion and KinectTitleButton. In GestureKeyboard the core element is Kinect Toolbox, that allows gesture recognition. Three default gesture - LeftHello,RightHello and HadsTogether from Kinect Toolbox were used in GestureKeyboard.

The 8pen is more interesting in terms of implementation. Each sector in 8pen keyboard is a simple WPF polygon object. The cursor is a WPF ellipse object. The coordinates of cursor are mapped to the coordinates of users right hand. But the coordinate system of kinect differ from coordinate system of WPF windows. To solve this problem the ScaleTo function were used. ScaleTo function is a part of small but powerful toolkit - Coding4Fun. [35] .

In order to determine whether cursor intersect with one of the sector or with center of keyboard, the Hit Testing in the Visual Layer were used. The purpose of the HitTest methods in the VisualTreeHelper class is to determine whether a geometry or point coordinate value is within the rendered content of a given object, such as a control or graphic element. [36]

The Circulat keyboard is probably the most complicated of all. As well as GestureKeyboard it uses basic gesture recognition from Kinect toolbox. But core element of GestureKeyboard is open source pie diagram from tutorial [37]. And as was mentioned before, to determing the size of each sector, TextPredictionAPI was used.

The rotation of arrow in circularKeyboard is also worse to mention.Rotation of arrow is bounded to coordinates of users right hand. To achieve that we have transform kinect

coordinates to coordinates of WPF windows, and that transform them to polar coordinates to get angle of arrow. for transformation between WPF coordinates and polar coordinates, the standard converting between polar and Cartesian coordinates were used.

Chapter 5

Evaluation and Testing

This chapter evaluates the mid-air text input methods that was developed in this thesis, and whose implementation and design was described in the previous chapter.The five implemented methods are compared to each other.The main metrics for quantitative testing are: error rate, entry rate and subjective rating. The qualitative testing was also made in form of user feedback and observations during test.

At the beginning of this chapter the experiment design will be described.Then the results of qualitative and quantitative testing will be discussed.

5.1 Experiment design

As was mentioned before the main goal of this thesis is not only implement, but compare 5 different mid-air text input methods. To achive this, series of usability test were performed. Tests were performed on 5 participants, but each of this participant was tested 3 times.

5.1.1 Experiment environment

The experiments were conducted in a room of size 4x8 m2 with one big window. There was enough space for user to move freely. The distance between participant and Microsoft Kinect was about 1,7-2 m, depending of participant height. Before the beginning of each test, the tilt of Kinect was adjusted to correspond users height. The most problematic part with setting up experimental environment, was simulation of big screen environment. Eventually, it was decided to use a television set with a diagonal 50" and with Full HD resolution (1280x1080px). The distance between participant and televisions set was also 2m. To cancel out the glare effect on a screen, all experiments were performed in a night hours after sunset.

During each test, the moderator was present.The main objectives of moderator were:

- Explain to participants how to use Kinect properly.
- Explain to participant how each of the mid-air text input method works and how to use it.
- To keep time during experiment.
- To switch sentence in time.
- Observe participant's behaviour and comments, during the experiment.

5.1.2 Experiment Scenario

One of the most important task in this testing, was to make sure that participant did not nervous too much and perform text input as closer to real situation as it possible. To achieve that moderator have explained to each participant, that we were not testing them but we were testing our software and that they can do any number of errors, and they can correct errors only if they want to.

Because test was designed as within-subjects an additional effect must be accounted: learning. For example, if two techniques, A and B, are compared and all participants used technique A first followed by B, then an improvement with B might occur simply because participants benefited from practice on A.The solution is counterbalancing 5.1 . [38] Because we were testing 5 different mid-air text input methods, and 5 people were participaiting in our tests, a Latin Square of size 5x5 was used to determin order of text input methods and conterbalancing for each participant.

qwerty	3push	gest	8pen	circular
3push	8pen	qwerty	circular	gest
8pen	curcular	3push	gest	qwerty
curcular	gest	8pen	qwerty	3push
gest	qwerty	circular	3push	8pen

Table 5.1: Latin Square for counterbalancing

The overall process of testing was quite straightforward. Each participant has have 10 minutes for each text input method,to transcribe phrase that was presented to him 5.1. Phrases was randomly selected from file that contains 500 different phrases.The time that it took to write the phrase, was measured from the moment when phrase appeared on the screen, to the moment when user writes last letter of the phrase.

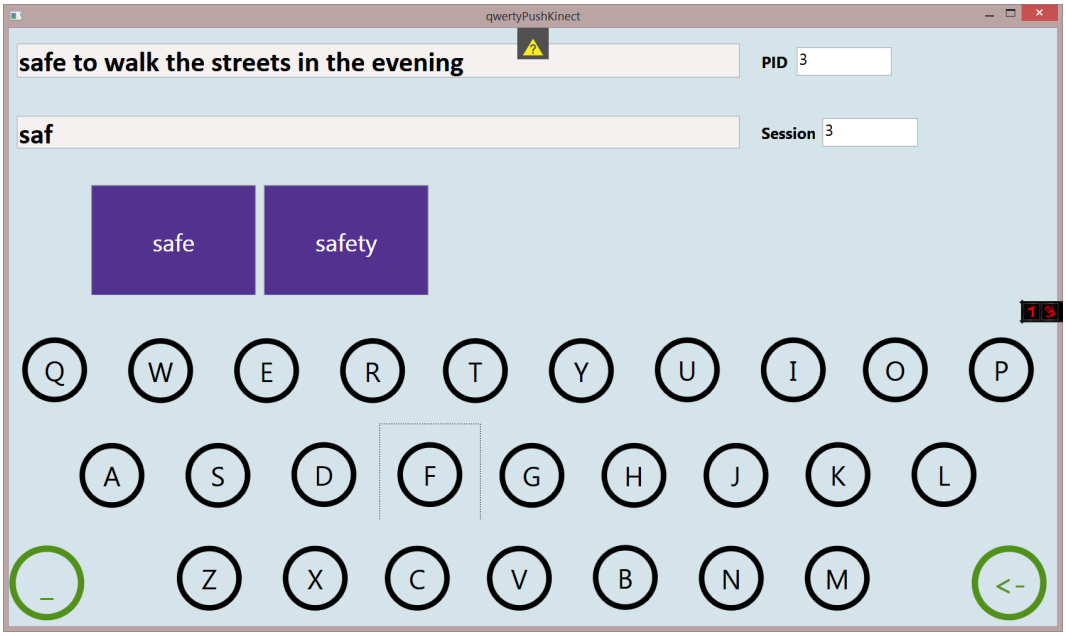


Figure 5.1: Testing phrase example.

5.2 Quantitative Results

During the testing, the mid-air text input application was logging different parameters of experiment which are:

- Presented phrase.
- Transcribed phrase.
- Number of backspaces.
- Number of buttons/unistrokes/gestures.
- Time it took to transcribe phrase in milliseconds.

This parameters were used to calculate WPM (Words Per Minute), corrected error rate, uncorrected error rate and total error rate.

Words Per Minute is one of the most common empirical measure of text entry performance. Usually a "word" is defined as sequence of 5 characters including space. It is worth to mention that WPM measure does not consider the number of keystrokes, keypresses or gestures made during text entry. WPM measures only the lenght of the resulting string and how long it takes to enter it. [38]

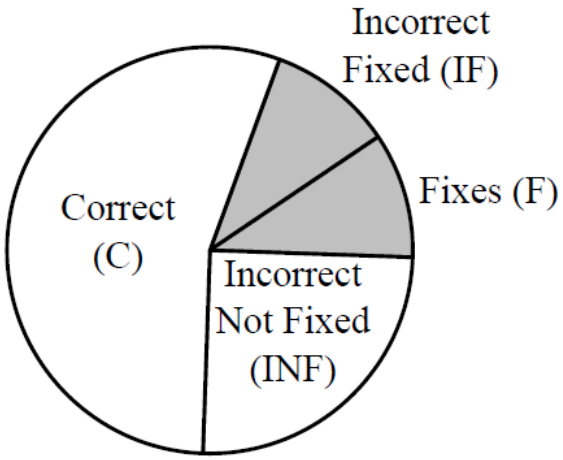


Figure 5.2: Input stream classes. [39]

Input stream that user produce contains not only characters that user enters, but also keystrokes and editing commands, like backspaces, detele commands, cursor movements and etc. Figure 5.2 divides the keystrokes of the input stream into four classes, depending on how they affect the error rate, where: [39]

- **Corrected (C)** - all correct characters in the transcribed text .
- **Incorrect-not-fixed (INF)** - all incorrect characters in transcribed text.
- **Incorrect-fixed(IF)** - all characters backspaced during text entry.
- **Fix(F)** - all backspaces.

Using this character classes three error rates can be defined:

$$correct\_error\_rate = (IF)/(C + INF + IF) \tag{5.1}$$

$$uncorrected\_error\_rate = (INF)/(C + INF + IF) \tag{5.2}$$

$$total\_error\_rate = (INF + IF)/(C + INF + IF) \tag{5.3}$$

The average results of each mid-air text input method in each session are shown in figure 5.3.Table contains values of WPM, correct error rate, uncorrected error rate and total error rate. For better visualization bar charts were used 5.4, 5.5, 5.6, 5.7.

The results of our research are very interesting indeed. As you can see 5.4, the overall WPM is not very high actually its quite low. The fastest text input method is QWERTY,

Session	Method	WPM	Corrected Error Rate	Uncorrected Error Rate	Total Error Rate
s1	QWERTY	3.456	0.071	0.007	0.078
s1	3push	3.096	0.127	0.007	0.133
s1	gesture	2.328	0.138	0.004	0.142
s1	8pen	2.381	0.153	0.019	0.171
s1	circle	1.97	0.111	0.003	0.114
s2	QWERTY	4.309	0.055	0.02	0.075
s2	3push	3.095	0.112	0.005	0.117
s2	gesture	2.781	0.09	0.021	0.111
s2	8pen	2.044	0.153	0.013	0.165
s2	circle	2.483	0.123	0.008	0.131
s3	QWERTY	4.116	0.056	0.007	0.064
s3	3push	3.575	0.097	0.01	0.108
s3	gesture	2.908	0.102	0.014	0.116
s3	8pen	2.612	0.135	0.013	0.148
s3	circle	2.608	0.098	0.008	0.107

Figure 5.3: Average results of each mid-air text input method in each session.

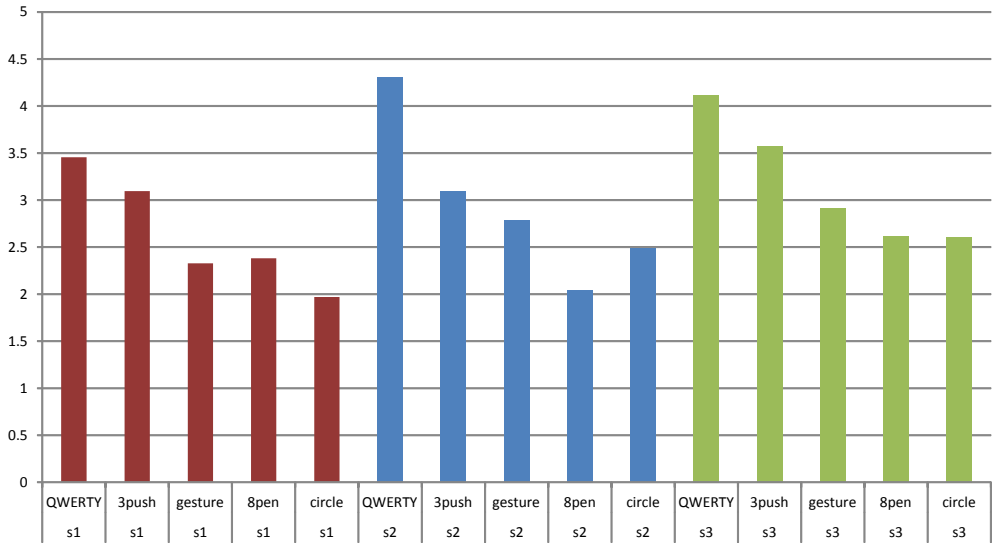


Figure 5.4: Average WPM results of each mid-air text input method in each session.

and as you can see from the bar chart, it's maximum performance is slightly more than 4 words per minute, but average result for the physical qwerty keyboard is 40 words per minute, and it's 10 times more than mid-air QWERTY. And there is a few reason for that. The first one, is that on physical keyboard users typing with all 10 fingers, and that means that user does not have to move cursor from one button to another. The second one is tactile feedback, when user types on physical keyboard, he can not only see buttons, but also feel them, but tactile feedback for mid-air software it's a good topic for future researches.Furthermore it is possible to see that WPM increases from session 1 to session 3.

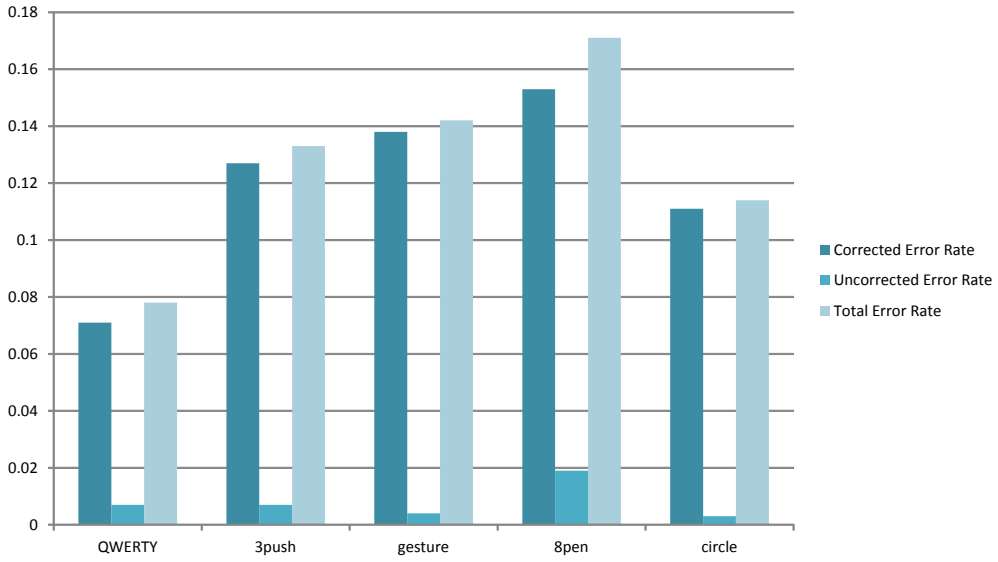


Figure 5.5: Error rates in session 1

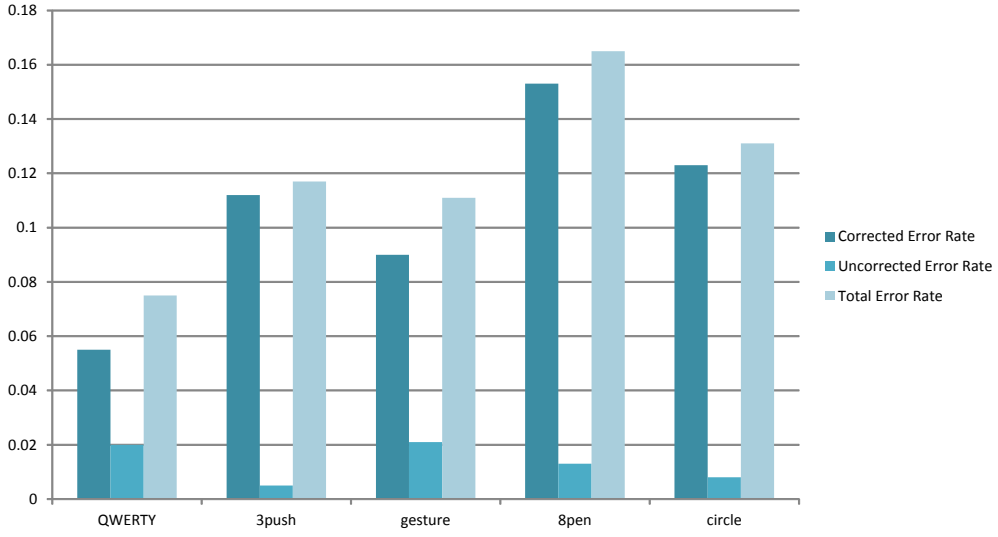


Figure 5.6: Error rates in session 2

That means that with more session we can achieve better result in WPM. On another hand the error rate of mid-air text input is quite low, this may be due to the fact that overal WPM is quite low too. In any case, main goal of mid-air text input is not replace physical keyaboard, but to help people write short phrases, in context of big screen enviroment. But lets look closer on each result.

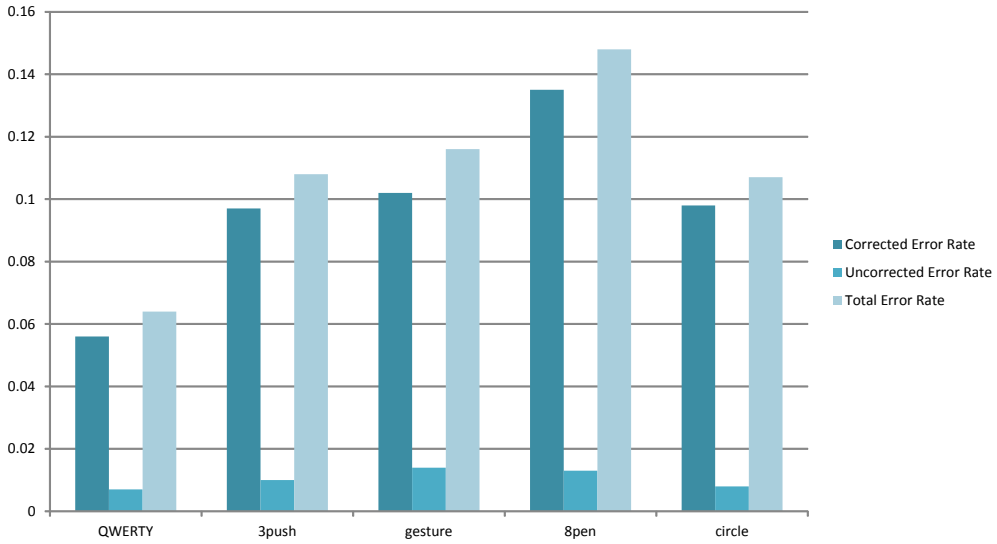


Figure 5.7: Error rates in session 3

As you can see from result, the mid-air QWERTY is the fastest method from all, moreover it has lowest correct error rate,uncorrected error rate and total error rate.The performance of mid-air QWERTY can be explained through the fact, that QWERTY layout familiar to everyone, who had use a standard PC keyboard. That means that participants did not spend much time, trying to find proper letter. Word completion also works great in this method. Participant have to type just a few letters from word, and then he was able to select word from list. Word completion can also explain low error rates. KinectTitleButton and KinectCircleButton also works great. Participant have learned very quickly how to use them.

Maybe QWERTY shows better result but the results of 3Push is much more interesting. The 3Push have quite good WPM, probably because it also uses KinectTitleButtons and Kinect CircleButtons. The layout of 3Push is also very simple - it has only 5 buttons. But error rates are quite high, probably because of the UniGlyph. Participants not always were able to track what they actually write, especially in case of long and complicated words.

As was mentioned before, Gesture mid-air text input is very similar to the 3Push, the only difference is that the each UniGlyph character is bounded to gestures. And that fact can explain low WPM of this method. It takes more time to perform LeftHello, RightHello and HandsTogether, then push gesture. Moreover Kinect Toolbox is not perfect, and sometime participant have to repeat gestures few time, before Kinect Toolbox actually recognize it. Perhaps better gesture recognition API and better gesture design can improve WPM of this method.The situation with error rates a quites the same as in case of 3Push. Again participant loses control of what they write.

The situation with WPM of 8penKinect text input method can be explained through the fact, that the layout of this method was completely unknown for all participants. An all participants were spending quite a lot of time to actually find letter. Also there was some technical issue with mid-air 8penKinect - when users hand was in front of his body sometimes cursor starts jiggling around. Cursor jiggling was slowing the process of text input and increases error rates.

The last text input method is Circle keyboard. The WPM of this keyboard is quite similar to the WPM of 8pen and Gesture keyboards, but error rate is lower. The situation with WPM can be explained through the fact that layout of Circulat keyboard is constantly changing, that means that user have to search for the letter again and again. And again there was problem with Kinect Toolbox - sometime participant have to repeat gestures few times, before Kinect Toolbox recognize it. Low error rate can be related with letter prediction.

5.3 Qualitative Results

In this section information, that was collected during observation and interviews with participants, will be summarized.

The QWERTY keyboard is probably the most simple to discuss. As was mentioned before, QWERTY layout was familiar to every participant, so they did not have any problems with understanding of how this text input method works. The only problem, that many of the participants were experiencing was push gesture. And the beginning most of them were pushing too far, sometimes they even were bending toward kinect. But after moderator have explained how to perform push gesture properly, participants did not experience problem with it any more. All participants agreed that QWERTY keyboard was good, but too slow. One of the participants mentioned that he have experienced problems with sticky effect of the KinectCircleButton. Sometimes cursor was sticking to another button during push gesture.

The situation with 3Push input method was quite good too. Participants have learned how to use UniGlyph very quickly.Some of the participant were experiencing the same problem whit push gesture as they were with QWERTY. But again, after a brief explanation everything were fine. 3 from 5 participants were complaining that they not always were able to track what they actually write, especially in case of long and complicated words. The majority of participants agreed that sometimes list of possible words is too long, and it takes some time to find proper word in this list. The main problem of Gesture keyboard was gesture recognition. All participants were complaining, that some time they have to repeat gesture few times, before application recognize it, especially in case of HandsTogether

gesture. The problems with gesture recognition were slowing text input and destructing participants. As well as in the case of 3Push keyboard, participants were experiencing problems with tracking of what they actually write. And the same problem with long list of possible words occurs in this text input method.

Participants opinions about 8penKinect were ambiguous. On one hand, participants have learned how to use 8penKinect very quickly. Most of them admitted that movement, that they need to perform to produce letter are very simple and natural. On the other hand, the layout of 8penKinect keyboard was completely unfamiliar for all participants. They were spending a lot of time just to find required letter. In addition 2 participants pointed that 8penKinect method is quite tiring, because it requires always to hold right hand in front of kinect.

The Circular keyboard received quite interesting evaluation from participants. Letter prediction worked great in this text input method. In many cases participants did not have to move pointer line, they just have to performed LeftHello gesture again and again, because sector with necessary letter was big enough. On other hand, it was very hard for participant to select letter with small sector size. And again gesture recognition was not working perfectly. 2 participants were complaining that this text input method requires a lot of concentration, and that this method is tedious.

Chapter 6

Conclusion

During this research different motion capture devices and techniques were analyzed. Base on this analysis, it was decided to use Microsoft Kinect, because Kinect is already publicly available, has a reasonable price and a great support from wide community and directly from Microsoft. Furthermore Kinect is markerless motion capture device, and hance does not requires an additional equipment and does not depend on light condition of laboratory.

A lot of different text input method were also analyzed.Base on this research 5 different mid-air text input methods, QWERTY, 3Push, Gesture, 8penKinect and Circular, were designed, implemented and tested. The mid-air QWERTY is based on classical virtual QWERTY keyboard. 3Push and Gesture are based on idea of UniGlyph.8penKinect is basicly mid-air clone of popular keyboard for touchscreen devices - 8pen. Circular keyboard is a improvment version of circular keyboard from article [3].

The result of testing are very interesting. They are suggested that the most suitable methods for adaptation are QWERTY and 3Push. And in my personal opinion these text input techniques can achieve even better results with more testing and improvements in design. For example it will be a good idea to replace push gesture with grip gesture, this gesture is much more faster and simpler. It is also worth to improve visualization of UniGlyph characters. The result of Gesture, 8penKinect and Circular are not so impressive but still these methods are valuable source of information for future researches . And in my opinion it's possible to achieve better result with these text input methods, but it is necessary to improve design of gestures, and to find a better gesture recognition API. In case of Circular keyboard it is also nessesary to improve the sized of sectors whith low probability. The two main problems of 8penKinect are cursor jiggling and a high level of tiredness. And if the first problem can be solved relatively simply, the second one is more complicated. Maybe a good options to start, will be to bind center of 8penKinect keyboard to user's position.

The word of mid-air techniques, Kinect and motion capture in general, is very interesting and full of new, exciting ideas. And i am sure, that a lot of new inventions will came from this word to our everyday life. These research have achieved all it's goals. And i hope that this thesis will be a valuable source of information for future studies.

Bibliography

[1] Microsoft, "Xbox official website," Accessed: 2014-05-08. [Online]. Available: <http://www.xbox.com/cs-CZ/Kinect>

[2] 8bitjoystick, "E3 2009 : Microsoft at E3 several metric tons of press releaseapalloza," 06 2009, Accessed: 2014-05-08. [Online]. Available: <http://www.codeproject.com/script/Articles/ArticleVersion.aspx?aid=442506&av=634657>

[3] G. Shoemaker, L. Findlater, J. Q. Dawson, and K. S. Booth, "Mid-air text input techniques for very large wall displays," in *Proceedings of Graphics Interface 2009*, ser. GI '09. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2009, pp. 231–238, Accessed: 2014-05-08.

[4] M. Bellis, "Eadweard Muybridge photo galler," Accessed: 2014-05-08. [Online]. Available: <http://inventors.about.com/od/weirdmuseums/ig/Eadweard-Muybridge/The-Horse-in-Motion.htm>

[5] MetaMotion, "Who uses motion capture?" Accessed: 2014-05-08. [Online]. Available: <http://www.metamotion.com/motion-capture/motion-capture-who-1.htm>

[6] K. Imran, "Simple and easy-to-use pie chart controls in wpf," Accessed: 2014-05-08. [Online]. Available: <http://www.codeproject.com/script/Articles/ArticleVersion.aspx?aid=442506&av=634657>

[7] MetaMotion, "Gypsy motion capture system workflow," Accessed: 2014-05-08. [Online]. Available: <http://www.metamotion.com/gypsy/gypsy-motion-capture-system-workflow.htm>

[8] P. Stolař, "Snímání pohybu pomocí systému Kinect," Master's thesis, Czech Technical University in Prague, Czech Republic, 2012.

[9] P. Nogueira, "Motion capture fundamentals," 2011, Accessed: 2014-05-08. [Online]. Available: <http://paginas.fe.up.pt/~prodei/dsie12/papers/paper_7.pdf>

[10] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella, "The process of motion capture: Dealing with the data," in *Computer Animation and Simulation '97*, ser. Eurographics, D. Thalmann and M. Panne, Eds. Springer Vienna, 1997, pp. 3–18.

[11] A. S. P. Ashish Sharma, MukeshAgarwal, "Motion capture process, techniques and applications," in *International Journal on Recent and Innovation Trends in Computing and Communication*, pp. 251–257, Accessed: 2014-05-08.

[12] J. Birn, "Jeremy birn reports from nab 2000 in las vegas." 2008, Accessed: 2014-05-08. [Online]. Available: <http://www.3drender.com/updates/NAB00_framecontent.htm>

[13] E. Price, "Leap motion launches its motion-controlled app store," 06 2013, Accessed: 2014-05-08. [Online]. Available: <http://mashable.com/2013/06/24/leap-motion-airspace/>

[14] Marc, "The unofficial leap faq," Accessed: 2014-05-08. [Online]. Available: <https://forums.leapmotion.com/forum/general-discussion/general-discussion-forum/434-the-unofficial-leap-faq>

[15] R. Aranda, "A look back at the nintendo wii," Accessed: 2014-05-08. [Online]. Available: <http://web.archive.org/web/20080212080618/http://wii.nintendo.com/controller.jsp>

[16] K. Castaneda, "Nintendo and pixart team up," 05 2006, Accessed: 2014-05-08. [Online]. Available: <http://www.nintendoworldreport.com/news/11557>

[17] K. Ohta, "Image processing apparatus and storage medium storing image processing program," Sep. 13 2007, uS Patent App. 11/522,997.

[18] ——, "Coordinate calculating apparatus and coordinate calculating program," Sep. 13 2007, uS Patent App. 11/405,664.

[19] A. Jana, *Kinect for Windows SDK Programming Guide*, ser. Community experience distilled. Packt Publishing, Limited, 2012.

[20] G. world records, "Kinect confirmed as fastest-selling consumer electronics device," Accessed: 2014-05-08. [Online]. Available: <http://community.guinnessworldrecords.com/_Kinect-Confirmed-As-Fastest-Selling-Consumer-Electronics-Device/blog/3376939/7691.html>

[21] T. STEVENS, "Kinect for windows sdk beta launches, wants pc users to get a move on," 06 2011, Accessed: 2014-05-08. [Online].

Available: <http://www.engadget.com/2011/06/16/microsoft-launches-kinect-for-windows-sdk-beta-wants-pc-users-t/>

[22] S. Crawford, "How microsoft kinect works," Accessed: 2014-05-08. [Online]. Available: <http://electronics.howstuffworks.com/microsoft-kinect.htm>

[23] P. Isokoski, "A minimal device-independent text input method," Tech. Rep., 1999.

[24] psbonline, "Virtual keyboard," Accessed: 2014-05-08. [Online]. Available: <https://www.psbonline.co.in/BankAwayRetail/(S(3izwzr554kvmhxbgks11su55))/web/L001/retail/jsp/user/vir_keyb.html>

[25] gate2home, "Qwerty keyboard," Accessed: 2014-05-08. [Online]. Available: <http://gate2home.com/English-Keyboard>

[26] K. Fiedlerová, "Possibilities of Text Input for Handicapped People," Master's thesis, Czech Technical University in Prague, Czech Republic, 2012.

[27] M. Belatar and F. Poirier, "Text entry for mobile devices and users with severe motor impairments: Handiglyph, a primitive shapes based onscreen keyboard," in *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, ser. Assets '08. New York, NY, USA: ACM, 2008, pp. 209–216.

[28] 8pen, "How it works," Accessed: 2014-05-08. [Online]. Available: <http://www.8pen.com/concept>

[29] P. Wilks, "Smart keyboard pro – t9, old-school," Accessed: 2014-05-08. [Online]. Available: <http://www.androidtapp.com/smart-keyboard-pro/smart-keyboard-pro-t9-old-school/>

[30] D. Grover, M. King, and C. Kushler, "Reduced keyboard disambiguating computer," Oct. 6 1998, uS Patent 5,818,437. [Online]. Available: <http://www.google.com/patents/US5818437>

[31] T. Huckaby, "Kinect for windows sdk 1.7 released, now includes kinect interactions & fusion features," Accessed: 2014-05-08. [Online]. Available: <http://devproconnections.com/development/kinect-windows-sdk-17-released-now-includes-kinect-interactions-fusion-features>

[32] dotneteers.net, "Kinect interactions with wpf - part i: Getting started," Accessed: 2014-05-08. [Online]. Available: <http://dotneteers.net/blogs/vbandi/archive/2013/03/25/kinect-interactions-with-wpf-part-i-getting-started.aspx>

[33] K. Toolbox, "Kinect toolbox documentation," Accessed: 2014-05-08. [Online]. Available: <http://kinecttoolbox.codeplex.com/documentation>

[34] P. Mička, "Letter frequency (english)," Accessed: 2014-05-08. [Online]. Available: <http://en.algoritmy.net/article/40379/Letter-frequency-English>

[35] Coding4Fun, "Coding4fun toolkit," Accessed: 2014-05-08. [Online]. Available: <http://coding4fun.codeplex.com>

[36] Microsoft, "Hit testing in the visual layer," Accessed: 2014-05-08. [Online]. Available: <http://msdn.microsoft.com/en-us/library/ms752097(v=vs.110).aspx>

[37] K. Imran, "Simple and easy-to-use pie chart controls in wpf," Accessed: 2014-05-08. [Online]. Available: <http://www.codeproject.com/script/Articles/ArticleVersion.aspx?aid=442506&av=634657>

[38] I. S. MacKenzie and K. Tanaka-Ishii, *Text Entry Systems: Mobility, Accessibility, Universality*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[39] R. W. Soukoreff and I. S. MacKenzie, "Metrics for text entry research: An evaluation of msd and kspc, and a new unified error metric," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '03. New York, NY, USA: ACM, 2003, pp. 113–120.

Appendix A

List of Abbreviations

GUI Graphical User Interface

API Application Programming Interface

UI User Interface

WPF Windows Presentation Foundation

SDK Software Development Kit

WPM Words Per Minute