Linnéuniversitetet
Kalmar Växjö

Master Thesis Project

New input methods for blind users on wide touch devices

Author: Andrii Krot
Supervisor: Dr. Ola Petersson
Reader: Dr. Narges Khakpour
Examiner: Prof. Welf Löwe
Semester: VT 2016
Course Code: 5DV01E
Subject: Computer Science

Abstract

Blind people cannot enter text on touch devices using common input methods. They use special input methods that have lower performance (i.e. lower entry rate and higher error rate). Most blind people have muscle memory from using classic physical keyboards, but the potential of using this memory is not utilized by existing input methods. The goal of the project is to take advantage of this muscle memory to improve the typing performance of blind people on wide touch panels. To this end, four input methods are designed, and a prototype for each one is developed. These input methods are compared with each other and with a standard input method. The results of the comparison show that using input methods designed in this report improves typing performance. The most promising and the least promising approaches are specified.

Keywords: input method, touchscreen, muscle memory, touch-typing, blind, text entry.

Publications

1. A. Krot and V. Kauk, "Засіб вводу тексту," UA Patent 90059, 2014.

Contents

1 Introduction

This chapter describes problems of using input methods by blind people, lists existing solutions and points to the shortcomings of existing approaches.

Blind people cannot enter text on touch devices using common input methods. They use special input methods that have lower performance (lower entry rate and higher error rate) [1]. However, they can use a classic QWERTY physical keyboard [2], because it gives a tactile feedback, which is utilized by muscle memory.

Bringing advantage of muscle memory from using classic physical QWERTY keyboards by blind users to wide touch panels (especially tablets) is the main goal of the project. The typing performance could be improved by using a dynamic keyboard layout, which adapts to a user by changing the position and the size of buttons according to user's touches, instead of static one.

1.1 Background

This section describes the background needed to understand input methods and an evaluation of them.

WPM (words per minute) is a measure of words typed in a minute. The word equals 5 characters [3], including spaces and punctuation, so, for example, "I'm writing a report" counts as 4 words. *Error rate* is a ratio of incorrect characters to total characters. Error rate could be represented as an uncorrected error rate (measures errors remaining in the transcribed string) or a corrected error rate (measures errors during entry) [3]. The project is concentrated on the uncorrected error rate because the corrected error rate is already measured as part of WPM (it takes time to correct an error). *Muscle memory* is a physiological adaptation of fingers (and some other parts of a body) to the repetition of specific movements [4].

It is important to split an input method to an abstract description, which is platform- and details-independent, and a software implementation. The first one is called *input method*. It describes how a text is entered using an informal description, algorithms, flowcharts or diagrams. The second one is called *input method editor* (IME) [5]. It is software which implements the input method.

Touch-typing is typing without using a sight contact with keys. The muscle memory is used to know their locations instead. The workflow of the most common style of touch-typing:

1. A user places eight fingers (no thumbs) on the home row keys. It is usually keys "A", "S", "D", "F" for fingers of the left hand and "J", "K", "L", ";" for fingers of the right hand for a classic QWERTY keyboard. Some users place thumbs on the "Space" key. Notches on "F" and "J" keys help users to place fingers on the home row keys without the visual contact.

2. The user makes movement by a finger to a key and press it. One key is related to exactly one finger, but one finger is related to a few keys. Mapping of keys to fingers is represented in Figure 1.1.1
3. The user returns all fingers to the basic position on the home row keys – step 1.

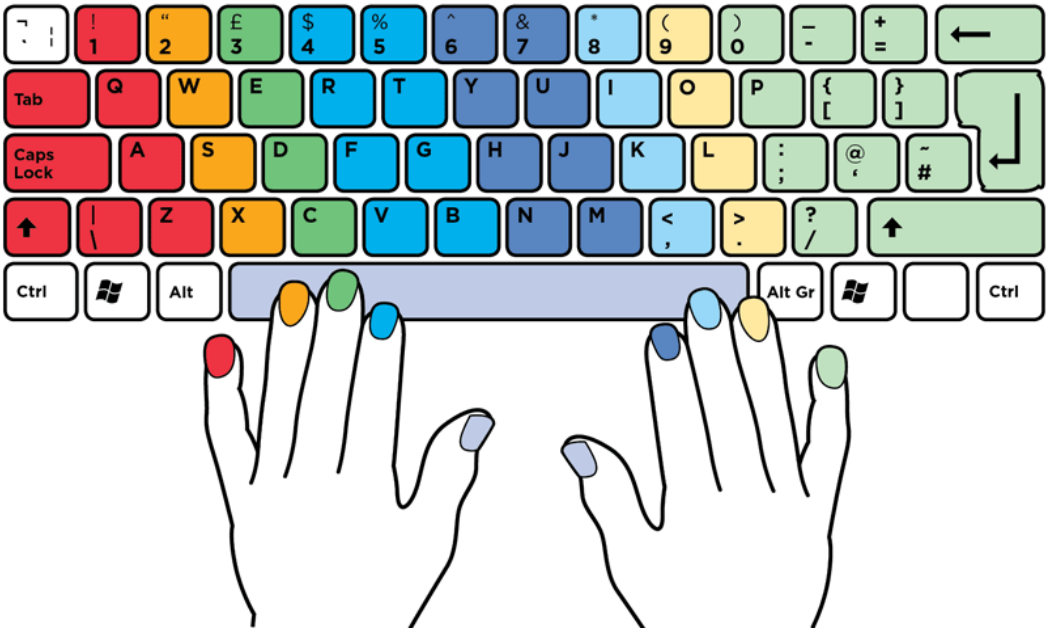

Figure 1.1.1: Typing zones for touch-typing on a classic QWERTY keyboard [6]. Keys, which are pressed by the same finger, have the same color.

1.2 Problem formulation

Default accessible keyboards for iOS (VoiceOver) and Android (TalkBack) are QWERTY-based input methods, so they use the user's knowledge of the QWERTY layout (order of buttons), but not the muscle memory from using QWERTY keyboards. The most probable reason is that using such skills requires wide touch panels, but tablets got popularity just a few years ago.

Scientific papers do not have enough information about taking advantage of this muscle memory to improve the typing performance of blind people on wide touch panels. Therefore, there are no prototypes for such input methods, no evaluation of them and no comparison to existing input methods for blind people.

1.3 Motivation

Since the number of old visually impaired people is much bigger than the number of young ones [7], most blind people was sighted before blindness.

Therefore, most blind people were working with standard QWERTY keyboards, and they have the skill of using such keyboards so that they can use a part of this skill even without the visual contact with the keyboard. It is especially actual for the touch-typing skill because touch-typing implies typing without the visual contact.

The average typing speed with the standard physical keyboard is 40 WPM [8], while a standard Apple touch input method for blind users gives just 0.66 WPM [1]. Therefore, usage even part of the skill of using physical keyboards could improve the typing speed palpably and make such input method demanded among bind users' society. Blind users' society is not the only one group that could be interested in the stated research problem; other groups are pointed in Section 1.6.

1.4 Purpose and research question

The purpose of the project is to develop an input method for blind users to type on touch devices utilizing the muscle memory from using physical QWERTY keyboards. The input method should be described as a normative knowledge; it should answer the question: "How to improve typing performance of blind people on wide touch panels utilizing the muscle memory from using physical QWERTY keyboards?"

Since the muscle memory can be utilized in many different ways, the solution is to design several input methods, develop a prototype for each one and compare them with each other and with the standard input method. The analysis of processed results of comparison answers the stated research question.

1.5 Limitation

The project is limited in order to be focused on the stated research problem and to avoid scattering over additional features. The purpose of the project is to develop the input method and the prototype with basic functionality: it should be possible to type lowercase characters of the Latin alphabet (a..z) and space, but not specific symbols, punctuation marks, numbers, uppercase symbols or symbols of other alphabets. It refers to evaluation as well: sentences for experiments should contain just lowercase symbols of the Latin alphabet and spaces. The reason for this limitation: since the width of the touch panel of the average tablet is less than the width of the classic QWERTY keyboard, the input method can contain the full-sized QWERTY layout, but not numerical keys, modifiers (like "Shift") and other keys. Nevertheless, the input method could implement special functionality for entering these buttons (for example, gestures), but this feature is not directly related to the muscle memory from using physical QWERTY keyboards, so it could distract evaluation results from the main idea (evaluation of additional functionality would be mixed with evaluation of basic functionality). Therefore, the project is concentrated on the

basic functionality. However, the possibility of implementing modifiers (buttons like “Shift”) should be considered, because it significantly expands abilities of an input method (even one modifier can double the number of supported characters).

Using the input method by deaf users is out of scope. Input methods that do not use an audio feedback and support blind users are possible (for example, using vibration), but it significantly decreases the performance. Since other input methods for blind users use audio feedback [1], [9], [10], [11], [12], comparing input method for deaf users with them would be unfair.

1.6 Target group

Researchers of input methods for both blind and sighted users could use this project to improve their own input methods or to create new input method based on the one from this project. Researchers of input methods for sighted users could be interested in using this project to apply the skill of touch-typing to touch devices.

Individual software developers and software development companies could develop software that implements input method of the project.

Tablets manufacturers (like Samsung) and OS developers (like Google and Apple) could integrate input method of the project with a default keyboard applications to improve the performance of using it by people with the skill of touch-typing.

1.7 Report Organization

The report is organized as follows. Chapter 2 gives an overview of the existing input methods. Input Methods are developed and described in Chapter 3. Software prototypes for developed Input Methods are developed in terms of Chapter 4. The evaluation of developed prototypes is performed by comparing them with each other and with a standard IME in Chapter 5. The conclusion is made in Chapter 6 based on evaluation outcomes.

2 Background

This chapter discusses previous input methods for blind users to type on touch devices and previous evaluations of them. The most popular input methods that are mentioned in scientific papers are listed in this chapter.

2.1 Braille-based input methods

Braille-based input methods require a user to know the braille writing system. Input methods of this type have six buttons; each button is related to one dot of the braille writing system. Figure 2.1.1 contains a sample of a character in the braille writing system. Some input methods have additional buttons that are mapped to actions like moving a cursor or characters removal.

Figure 2.1.1: Representation of the character ‘r’ in the braille writing system [9].

The methods are divided into two subtypes: chorded and non-chorded. Chorded ones require a user to press buttons simultaneously to produce a character; non-chorded ones allow the user to press buttons of the character with some time interval.

Typing speed of chorded input methods is low during first sessions, but it grows up after training. Perkinput is a chorded braille-based input method; it requires a user to make two touches to type a character. The first touch inputs left three dots of the character’s representation in the braille writing system; the second touch inputs right dots. An example of typing a character with Perkinput is represented in Figure 2.1.2. Perkinput’s entry rate is 4 WPM for the first session (it is a low rate, it almost equals the VoiceOver rate), but entry rate becomes nearly 10 WPM after 13 sessions, which is twice faster than VoiceOver [9].

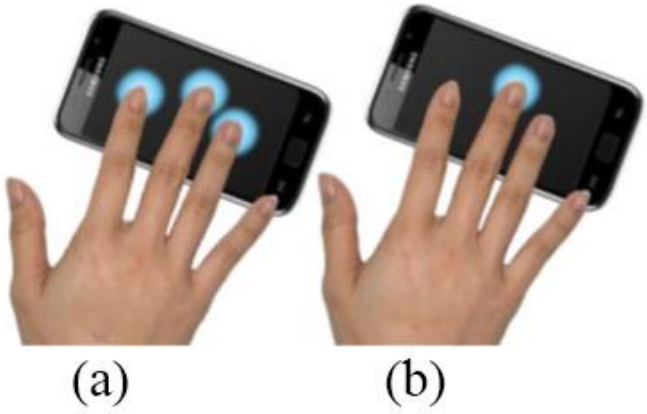

Figure 2.1.2: Process of typing character ‘r’ with Perkinput [9]. The user performs the touch “(a)” first and then the touch “(b)”.

Non-chorded input methods have opposite learning curve because the user does not have to learn it. Therefore, non-chorded input methods give high typing speed even during the first session. BrailleType is an example of the non-chorded input method. The touch panel is divided into six areas according to the template of the braille writing system (2 columns, 3 rows) for this input method. The user touches required areas with a single finger, so the amount of touches needed to type a character equals to the amount of dots in the representation of the character in the braille writing system. Figure 2.1.3 represents the process of typing a character with BrailleType. BrailleType’s entry rate is 2.11 WPM, which is faster than VoiceOver entry rate (1.45 WPM) in the same experiment [10]. However, the entry rate does not grow up that much after training.

Figure 2.1.3: The user of BrailleType has just typed character ‘r’ [10].

2.2 Stroke-based input methods

A user has to draw some figure using a touch panel to type a character by a stroke-based input method. Stroke-based input methods are divided into two subtypes.

Input methods like Unistrokes use strokes that differ from regular handwritten letters (Figure 2.2.1), but trained users can draw these strokes faster than normal handwritten letters (it leads to higher WPM). Moreover, it is easier for software to recognized the strokes (it leads to lower error rate and correction rate). The user draws a straight line to input frequent characters (e.g., E, A, T, I, R).

Input methods like Graffiti use strokes that are similar to regular handwritten letters (Figure 2.2.1), so such input methods are usable even during the first session. Nevertheless, some strokes are problematic for recognition by software. For example, "character 'O', 'T', 'E' and 'N' faces recognition problem" [1].

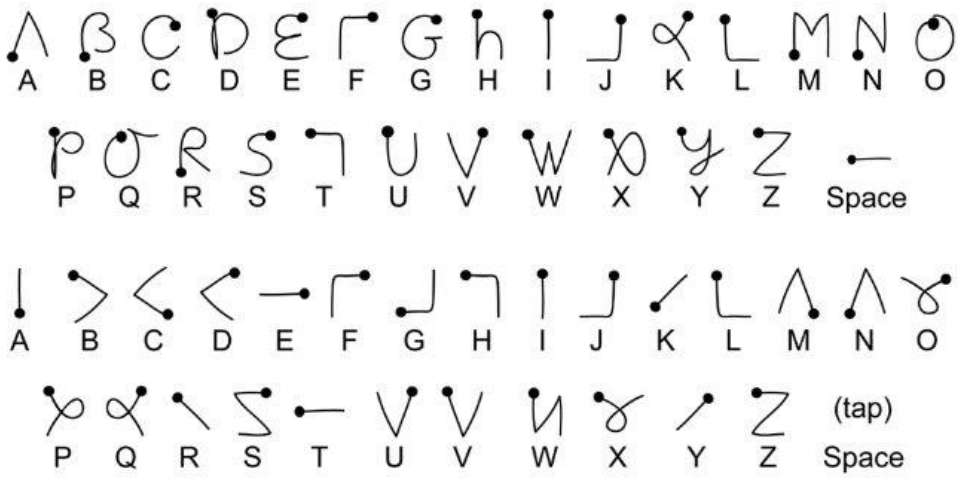

Figure 2.2.1: Graffiti alphabet (top) and Unistrokes alphabet (bottom) [13].

Therefore, input methods like Graffiti give better results (higher WPM and lower error rate) for first sessions, but input methods like Unistrokes give better results after training [13].

2.3 Telephone-keypad-based input methods

Telephone-keypad-based input methods were popular before the emergence of touch devices because it was the only option for phones like Nokia 5110 (Figure 2.3.1). In comparison with touch panels, these keyboards give the ability to feel a button's shape (it is of particular importance for blind users), but limit input to some number (usually, 12) of buttons. Telephone-keypad-

based input methods work on touch devices as well, but it leads to the problem with buttons locating because buttons of touch panels do not have a three-dimensional shape.

Figure 2.3.1: A keypad of Nokia 5110 [14].

Telephone-keypad-based input methods are divided into two subtypes. Input methods like MultiTap require a user to tap one button for a few times to type one character (for example, the user has to press button “5” twice to produce ‘K’). It leads to low entry rate but allows typing any desired sequence of characters.

Some input methods, for example, SVIFT, uses systems like T9, so to type a word with N characters a user has to press N buttons and a delimiter button and select the desired word from the list of predicted words [15]. For example, to enter the word “bet” a user presses buttons 2, 3, 8 and selects the desired word (“bet”) from the list of “bet” and “aft” (“aft” can also be made out of letters of 2, 3 and 8) words. This approach allows typing common words easily because they will likely be first in the list. However, this approach leads to problems with typing rare words: a user presses buttons in T9-style, try to find the desired word in the list of suggested words, cancel it and type it again using an input method like MultiTap.

2.4 QWERTY-based input methods

QWERTY-based input methods use the standard QWERTY keyboard layout. Since the number of old visually impaired people is much bigger than the number of young visually impaired people [7], most blind people was sighted before blindness. Therefore, most blind people were working with the standard QWERTY keyboards, and they have the skill of using such keyboards, so QWERTY-based input methods work especially well (in terms of higher WPM and lower error rate) for this group of people.

An example of a common scenario of using an existing accessible QWERTY-based input method: if a user slides a finger around a touch panel, the system pronounces each hovered key; the user makes some special action when the desired character is pronounced.

VoiceOver is the default iOS accessible keyboard. An example of the scenario of using VoiceOver (iOS 8): if a user touches a button, the system pronounces the character of the button; to type the character the user has to double-tap related button [13]. Therefore, if the user hovered a wrong button, he (or she) just has to slide the finger to another button or lift the finger up.

TalkBack is the default Android accessible keyboard. It is similar to VoiceOver, but to type a character a user just has to lift up the finger. Therefore, if the user does not want to type the character, he (or she) has to move the finger outside the keyboard.

SpatialTouch is similar to described QWERTY-based input methods (it is especially similar to TalkBack: a character is inserted by lifting the finger on the button), but it uses the left audio channel to pronounce characters of the left half of the keyboard and the right audio channel to pronounce characters of the right half of the keyboard. Different voice genders are used to enhance speech intelligibility. It allows the user typing by two fingers simultaneously. However, the evaluation has shown that this input method does not improve performance palpably [12].

3 Input Methods Design

This project is not just about the research of existing input methods, but also about developing new input method and researching of it. This chapter describes the process of the input method design. This chapter states an overview of input methods design, formalizes four different input methods and makes a summary of them.

3.1 An overview of design

Input methods described in Chapter 2 do not use the muscle memory from using physical classic QWERTY keyboards. Since blind people use such keyboards to type on PC [2], this potential is not utilized for typing on touch devices.

Users use the home row keys for orientation during touch-typing. Each finger lies on the personal key of the home row (space button is exception – it is covered by both thumbs) and responsible for striking a certain set of buttons [16]. The process of touch-typing could be simplistically described by the Figure 3.1.1.

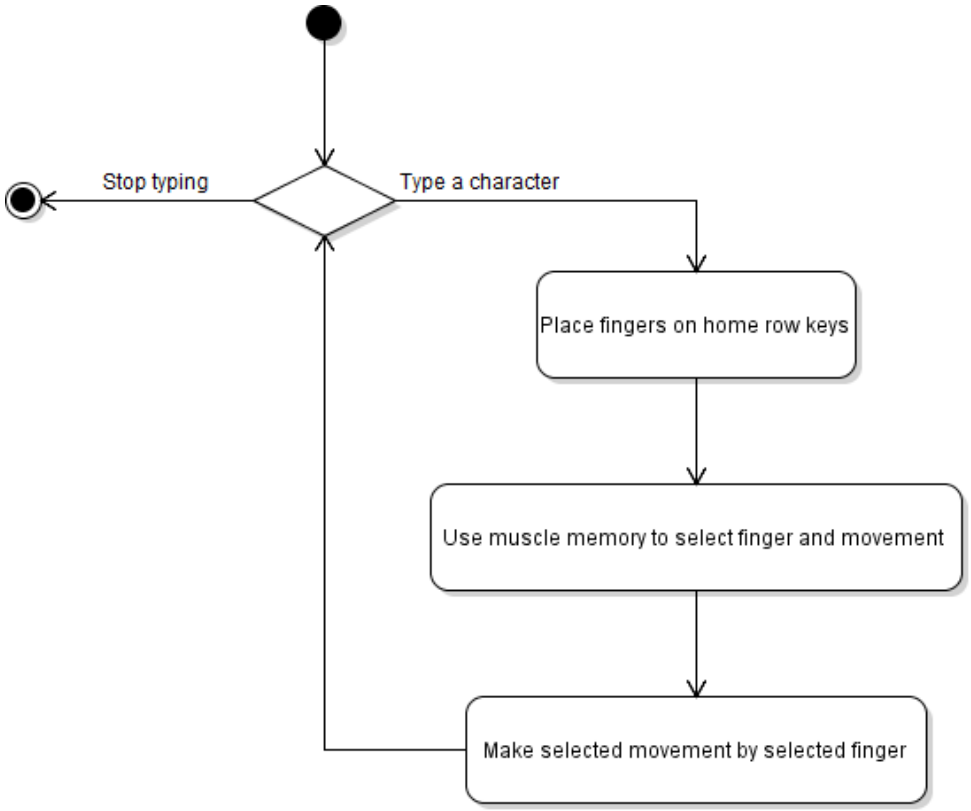

Figure 3.1.1: UML Activity Diagram of touch-typing.

Considering Figure 3.1.1, the user performs following actions during touch-typing:

- Decision “Is typing required” is made in user’s mind, so it does not depend on a keyboard.
- Action “Place fingers on home row keys” depends on the keyboard, because the user uses the shape of buttons for orientation.
- Action “Use muscle memory to select finger and movement” is made in user’s mind, so it does not depend on the keyboard.
- Action “Make selected movement by selected finger” depends on the keyboard in a unobvious way, so it is researched below.

Simple, informal experiment is performed to understand how much action “Make selected movement by selected finger” depends on the keyboard. A classic QWERTY layout was printed on A4 paper, the distance between centers of keys was standard (19.05 mm [17]). A camera was directed to the paper and was recording 60 fps video. Three subjects (all of them have the skill of touch-typing) were asked to put fingers on the home row keys, close eyes and type a simple sentence. Recorded video was manually processed so that ratio of correct touches to total touches is measured. The process of the experiment is shown in Figure 3.1.2. The experiment has demonstrated that subjects were touching right keys in most cases, so the action “Make selected movement by selected finger” depends on the keyboard, but not much, and it is possible to use a flat surface instead of a keyboard. Process and results of the experiment are not officially documented, because of two reasons:

- The experiment was performed as the first step of the project, because if the experiment showed opposite results, the project would have much less practical and scientific usage and it would be abandoned.
- The enhanced experiment will be performed when the prototype of the input method will be ready. Software prototype allows calculating parameters like WPM and error rate automatically, instead of manually video processing. Automatic processing is more precise and less time-consuming.

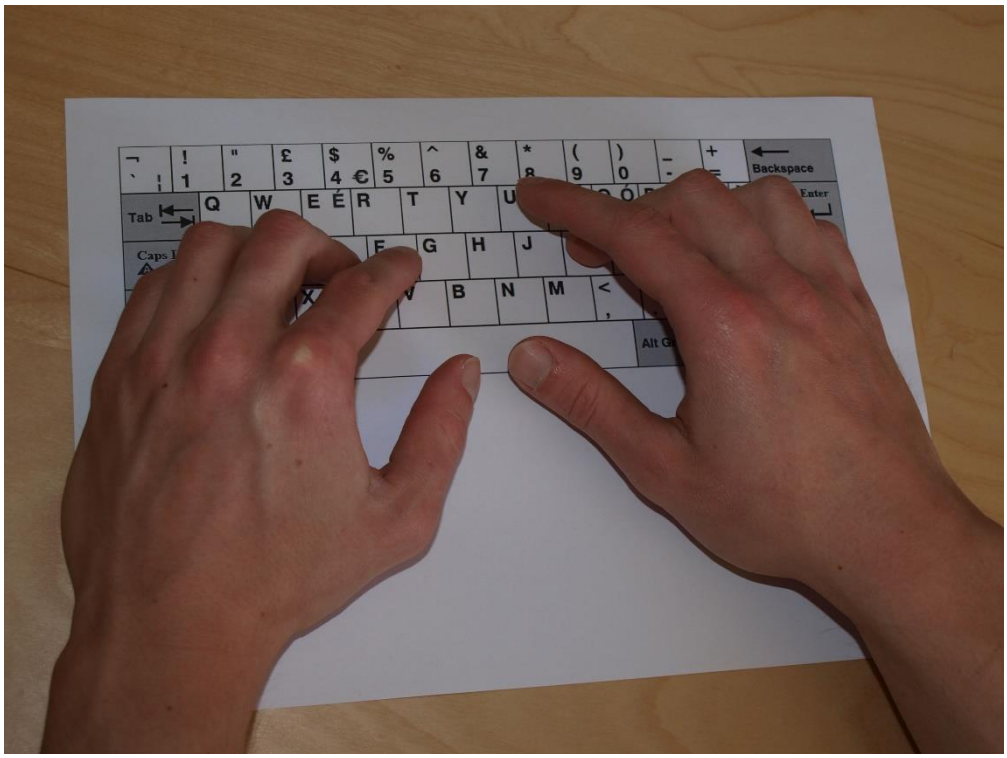

Figure 3.1.2: Process of the initial, informal experiment.

Therefore, a keyboard just has to help a user with the action “Place fingers on home row keys”. Users can be divided into two types: the ones who keep fingers laying on the home row keys between strokes and the ones who hold fingers in the air above the home row keys. Different kinds of behavior require different input methods because the muscle memory automatically returns fingers to the accustomed position.

Since most touch panels do not support differentiation of the “resting” touch from the “tapping” touch by default, the second style of touch-typing (holding fingers in the air above buttons during rest) is selected for the first input method, because this style does not have a “resting” touch and all touches should be recognized as “tapping” ones. Therefore, the main task of the developed input method is to assign fingers positions to buttons positions. In other words, the system has two objects (a finger and a button), and the system wants to be sure that the finger is located on top of the button. It could be done in two ways: “move” the finger or move the button.

To “move” the finger the system can give feedback to the user so that the user knows that the finger should be relocated. Common ways for a tablet to provide feedback to a user are a visual way (displaying some information on a screen) and a sound way (producing some sound). A tactile way (using vibration) is not taken into account, because some tablets, especially cheap

ones, do not have the vibration feature. The visual way is not suitable for this case because the target audience of the developed input method is blind users. The sound way is already patented [18]. Therefore, the input method should move the button, but not the finger.

To move the home row buttons relative to user's fingers system has to make three decisions:

- All buttons should be static relative to each other or the keyboard should be divided into few independent blocks? A block contains buttons, which are static relative to each other, but blocks can be moved and scaled independently. Therefore, the question sounds more like "What is the optimal number of independent keyboard blocks?"
- Which fingers should be used as points of orientation for keyboard blocks?
- When keyboard blocks should be moved according to fingers position?

According to David Rempel [19], users prefer the Microsoft Natural Elite keyboard among keyboards with different split and different split angle. The layout of Microsoft Natural Elite (Figure 3.1.3) is divided into two blocks, each block is related to one hand, buttons in a block are fixed relative to each other, so the developed input method should contain two blocks. Since home row keys of each block of the keyboard lie on a line, the developed input method should use pointer and little fingers of each hand to specify the position of each block (if middle and ring fingers will be utilized as well, the home row buttons will be located higher).

Figure 3.1.3: Microsoft Natural Elite keyboard.

Keyboard blocks can be assigned to fingers position under different circumstances, depending on user's touch-typing style:

- If the user holds fingers in the air above the home row keys during typing, then keyboard blocks should be assigned when all 10 fingers touch the panel. If the number of touch points is less than 8 (number of fingers on the home row), then all touches should be recognized as key pressing and should lead to characters entry.
- If the user keeps fingers laying on the home row keys between strokes, then keyboard blocks should be assigned when all 10 fingers touch the panel or 8 fingers (without thumbs) touch the home row keys, but not any other keys.

Therefore, different modifications of the input method could lead to different results depending on the user's touch-typing style, so the development of one input method is not enough – at least two input method should be developed.

3.2 Input method Simple-Touch

The first designed input method is *Simple-Touch*. In this input method, users hold fingers in the air above the home row keys. A description of this input method is given as follows:

- The keyboard contains two blocks.
- A block contains buttons, which are fixed relative to each other.
- Blocks can be moved and scaled independently to each other.
- The left block contains buttons that are related to the left hand (the ones that the user should press by fingers of the left hand in the ideology of classic touch-typing), the right block – buttons that are related to the right hand.
- The input method pronounces character that is typed.
- Initialization of blocks requires 10 touch points. Two lowermost points (points with largest Y value) are ignored since they are related to space button. Remaining points are sorted by the distance to the left side of the touch panel (the first point is the point with least X value). The first point is used as the left point of orientation of the left block, fourth point – the right point of orientation of the left block, fifth – the left point of orientation of the right block, eighth – the right point of orientation of the right block. Buttons of orientation are buttons that lay under pointer and little fingers of each hand. Therefore, the button under little finger of the left hand ('A' key) is used as the left button of orientation of the left block, the button under pointer finger of the left hand ('F' key) – the right button of orientation of the left block, the button under pointer finger of the right hand ('J' key) – the left button of orientation of the right block, the button under little finger of the

right hand (';' key) – the right button of orientation of the right block. Initialization is the process of adjustment of position, scale and tilt angle of each block to make center of each button of orientation matches related point of orientation.

- If the number of touch points is 10, the input method performs the initialization of blocks.
- A touch is ignored if it occurs between a touch with 10 touch points and a touch with 0 touch points. Therefore, the user can safely lift fingers up after the initialization of blocks without invocation of characters entry.
- In other cases, the input method types character if related button was touched and released.

The above input method allows pressing two buttons simultaneously, so modifier buttons (like "Shift") could be implemented in a classic way, and it is active while the user is holding it.

Since blind users cannot see the position of blocks, they do not know if the blocks are overlapping each other or partially gone out of the touch panel. Some of tablets do not have vibration function, so a possible solution to give the feedback is a sound. The idea of using stereo sound as the feedback for a blind user, where the left channel is related to the left part of the keyboard and the right channel is related to the right part of the keyboard, is inspired by the publication "TabLETS Get Physical: Non-Visual Text Entry on Tablet Devices" [12]. Therefore, the left audio channel of the developed input method should give information about the position of the left block, same to right one.

Simple beep sound is selected as the sound for feedback, because it is informative enough, simple enough not to distract user's mind and it makes the user feel that something goes wrong, and it should be fixed. It is not sufficient just to play the beep sound of the same volume when a block is located at a wrong place, because it does not give information to the user about how to fix the position. Therefore, the volume of the sound is directly proportional to the size of overlapped or gone part of the block. Additional advantage: it is logical for the user, because the volume is 0 (no sound) if the block is located at the correct position. Therefore, the description is extended by following statements to supply the user with information about position of blocks:

- If the left block is overlapped or gone out of the touch panel, the input method produces the beep sound from the left audio channel. The volume of the sound is directly proportional to the size of overlapped or gone part of the block. Same to the right block and the right audio channel.
- The input method pronounces entered characters.

3.3 Input method variants

Simple, informal evaluation of the input method Simple-Touch has shown that subjects feel uncomfortable during typing, because, since eyes of subjects are closed, they do not know about the position of fingers after initialization (when the fingers are raised up). Therefore, the subjects are afraid to acidentelly move the fingers during typing. This problem is not critical according to the evaluations (see Section 5.2), because users can understand if fingers are still located at a right position after typing a character using sound feedback, and they can fix the position by reinitializing blocks. However, feedback from subjects led the project to the idea of another input method, which considers these comments.

3.3.1 Input method Vary-Touches

Input method *Vary-Touches* is an input method that allows users to keep fingers laying on the home row keys between strokes. Therefore, the user keeps all 10 fingers on the screen for initialization of blocks; the user presses the button by the finger, keeping another 9 fingers touched, for character entry. This way the user does not afraid that fingers do not match buttons, because the input method always has access to the position of the fingers and it can correct the position of the blocks according to the fingers.

Described scenario does not allow pressing two buttons simultaneously, so modifier buttons (like “Shift”) should be implemented like switches: the first press activates a modifier and the second one deactivates it. For example, “Shift” works like a “Caps Lock” in this way.

The general scenario of touch-typing with this style:

- To type a character of the home row, a user increases pressure on related button until a keyboard gives feedback. Then the user returns the finger to the original position by decreasing pressure.
- To type a character of a non-home row, the user lifts a related finger, moves it to related button, puts the finger down and increases pressure on the button until the keyboard gives feedback. Then the user returns the finger to the original position by lifting it up, moving it and placing to the home row key.

The second statement can be implemented on a touch panel just as it is. The first statement requires adaptation, because the ability of pressure determination is not precise enough on most modern tablets, so the input method cannot be sure if the user is pressing the button or just having a rest on the button. One possible solution is using accelerometer or gyroscope, but this approach is already patented by Dryft [20]. Therefore, the following solution is selected: to type a character of the home row, a user lifts the finger up and places it down again.

Therefore, the normative description of the input method Vary-Touches is same as the normative description of the input method Simple-Touch, but last three statements (the ones related to touch processing) are changed to following ones:

- The input method has a state.
- When the input method comes to a new state, it checks if any of outgoing transitions are possible. If so, it immediately goes to the next state.
- "Typing the character" is a sub-state. When the input method comes to this state, it types the character of the last untouched button and immediately goes to the next state.
- The input method performs blocks initialization while the state is "Initialization".
- The transition "N touches" requires a touch with N touch points. If none of outgoing transitions is possible, the input method waits for next touch (when the set of touch points or their coordinates is changed) and checks again.
- The transition "all touches are on home row" requires all eight home row keys to be touched.
- Possible states: Non-initialized, Initialization, Releasing, Tapping, Typing the character.
- UML state diagram of the input method is shown in Figure 3.3.1.

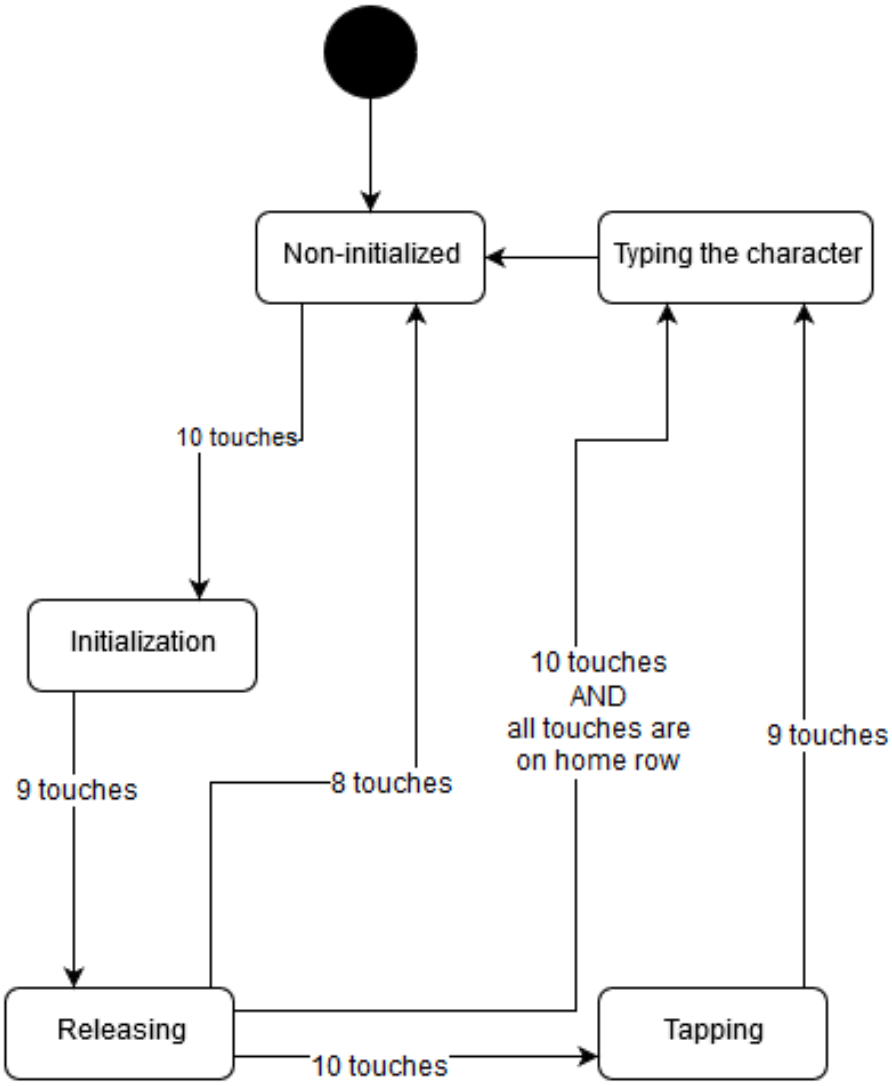

Figure 3.3.1: UML State Diagram of the input method Vary-Touches.

Simple evaluation has shown that subjects fell confused sometimes because of the inconstancy of typing, because a user has to make one tap to type a character of the home row, but two taps to type a character of a non-home row. Therefore, two other input methods are designed to overcome this problem: (1) the input method One-Touch that requires one touch to type a character, and (2) the input method Two-Touches that requires two touches to type a character.

3.3.2 Input method One-Touch

Input method *One-Touch* performs initialization whenever the number of touch points is 10. If the number of touch points drops from 10 to 9 and then returns

to 10, the input method types character of the last touched button. Therefore, the normative description of the input method One-Touch is same as the normative description of the input method Vary-Touches, but the input method is driven by UML State Diagram that is shown in Figure 3.3.2 and has four possible states: “Non-initialized”, “Initialization”, “Releasing”, “Typing the character”.

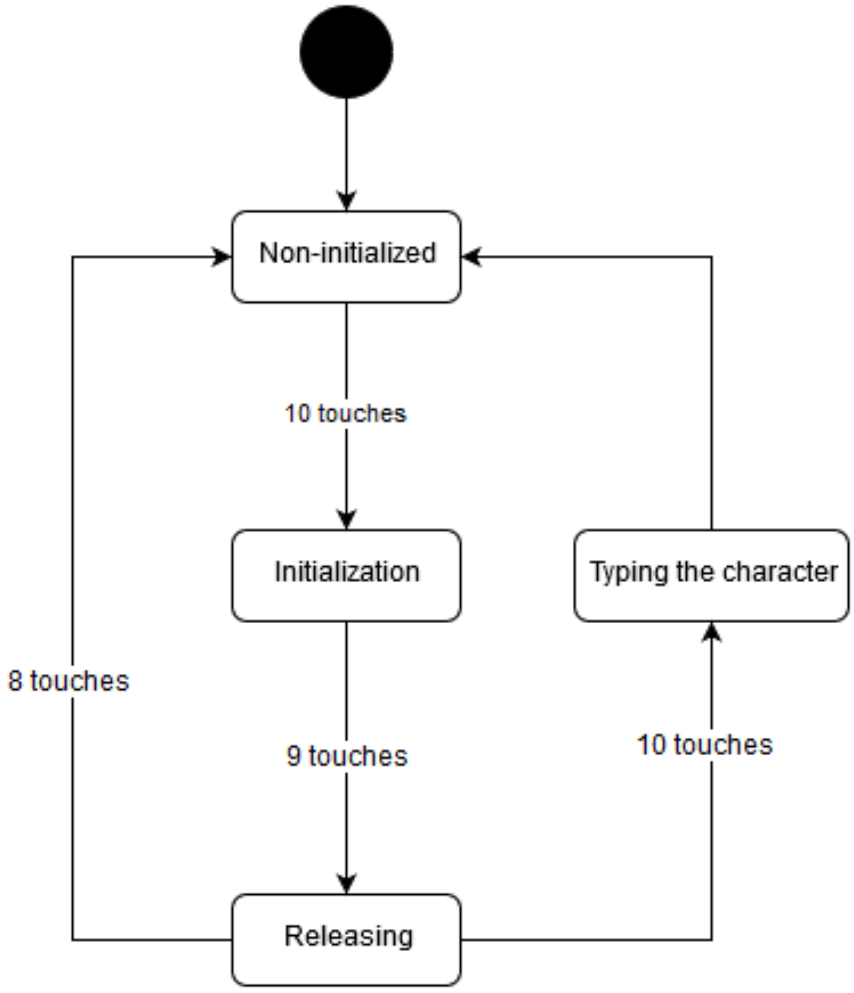

Figure 3.3.2: UML State Diagram of the input method One-Touch.

3.3.3 Input method Two-Touches

Input method *Two-Touches* stores a history of numbers of touch points. It waits for the pattern “10-9-10-9” (numbers are numbers of touch points). The input method types a character whenever the last step (last “9” of the pattern) occurs. It is initializing blocks during the first step (first “10” of the pattern). In other words, it is same as the input method Vary-Touches, but without direct

transition from the state “Releasing” to the state “Typing the character”. The normative description of the input method Two-Touches is same as the normative description of the input method Vary-Touches, but the input method is driven by UML State Diagram that is shown in Figure 3.3.3 and has five possible states: “Non-initialized”, “Initialization”, “Releasing”, “Tapping”, “Typing the character”.

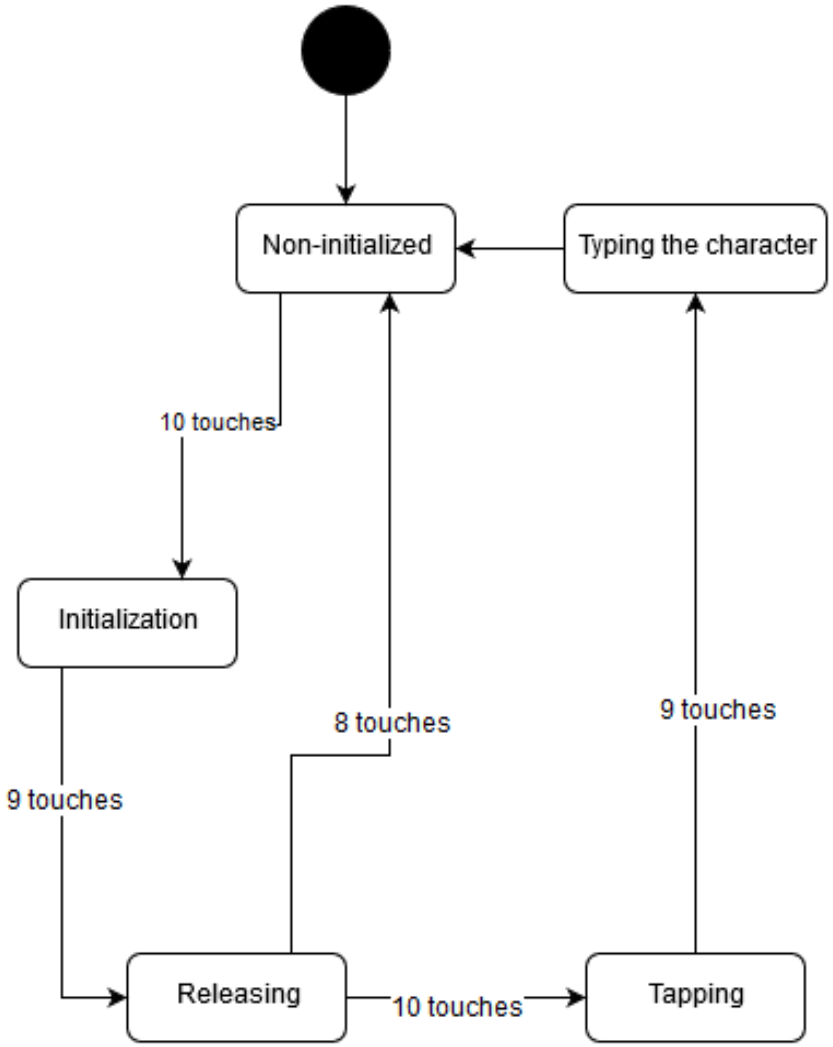

Figure 3.3.3: UML State Diagram of the input method Two-Touches.

3.4 Input methods summary

A brief summary of developed input methods is provided in Table 3.4.1.

Input method	How to type a character	Reason for the name
Simple-Touch	0. Place 10 fingers on a touch panel and move them until a beep sound disappear. 1. Lift all fingers up. 2. Place the finger on the button. 3. Lift the finger up.	The process of typing is similar to the process of typing on classic keyboards.
Vary-Touches	0. Place 10 fingers on a touch panel and move them until a beep sound disappear. 1. Lift the finger up. 2. Place the finger on the button. 3. If the button is not a home row one, lift the finger up and place to the home row.	The user makes 1 touch for typing a character of the home row, but 2 touches for a non-home row character.
One-Touch	0. Place 10 fingers on a touch panel and move them until a beep sound disappear. 1. Lift the finger up. 2. Place the finger on the button. 3. If the button is not a home row one, move the finger to the home row.	The user makes 1 touch for typing a character.
Two-Touches	0. Place 10 fingers on a touch panel and move them until beep sound disappear. 1. Lift the finger up. 2. Place the finger on the button. 3. Lift the finger up and place to the home row.	The user makes 2 touches for typing a character.

Table 3.4.1: Brief summary of developed input methods.

Figure 3.4.1 shows the difference between developed input methods.

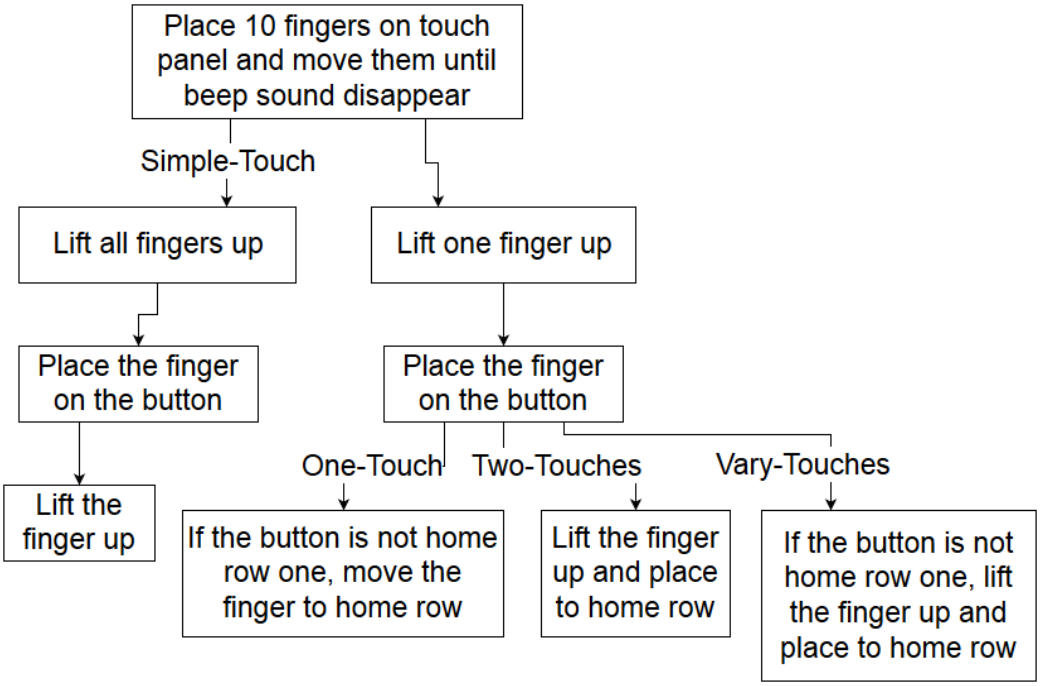

Figure 3.4.1: Difference between developed input methods.

4 Implementation

Software prototypes of the designed input methods need to be developed in order to perform an evaluation. Each prototype can be called an input method editor (IME) according to Section 1.1. This chapter discusses the development of these prototypes. This chapter covers requirements, architecture and coding of each prototype and describes the developed application.

4.1 Requirements

The prototypes should implement the normative description of input methods described in Chapter 3. The prototypes should work in the same scope and limitations as input methods.

Android is selected as a platform for prototypes, because of two reasons:

- Android allows using 3rd party keyboards so that users could use the developed application in real life after some modifications. iOS allows using 3rd party keyboards as well [21].
- As already mentioned, developed input methods require a wide screen. iPad Pro has big enough screen of 12.9 inches [22], but it was not available during the evaluation. Some Android tablets have wide enough screen. It is calculated later in the section.

Moreover, selected device should be able to run the prototypes. The device should work with Android OS, have a touch screen (because the input methods are based on touches processing), allow working with 10 touches simultaneously and have a wide enough touch screen. Tablets with 10 inches screen and 16:10 screen ratio are suitable because the width of the screen for this case is 21.5 cm. Since the standard distance between centers of keyboard keys is 19.05 mm [17], the total width of 11 keys (keys that contain characters of the Latin alphabet) is 20.955 cm. Therefore, 10 inches tablets have wide enough screen. ASUS TF700T fits these requirements, so it is selected as the device for experiments.

4.2 Architecture

The main requirement for the architecture is that it should allow using different modifications of an input method without duplicating the code. Moreover, the architecture is developed with taking into account additional features that are out of the scope of this report, but will be most likely added if the application will be used in real life.

The architecture of the developed prototype is shown in Figure 4.2.1. The diagram is simplified; most entities are dropped to concentrate attention on the most important solutions. Since the main goal of the diagram is showing the architecture in a very abstract way, it is not a pure UML Class Diagram: methods, fields, and package names are dropped; a class and an interface are

generalized to same entity; a package notation is used to describe external libraries.

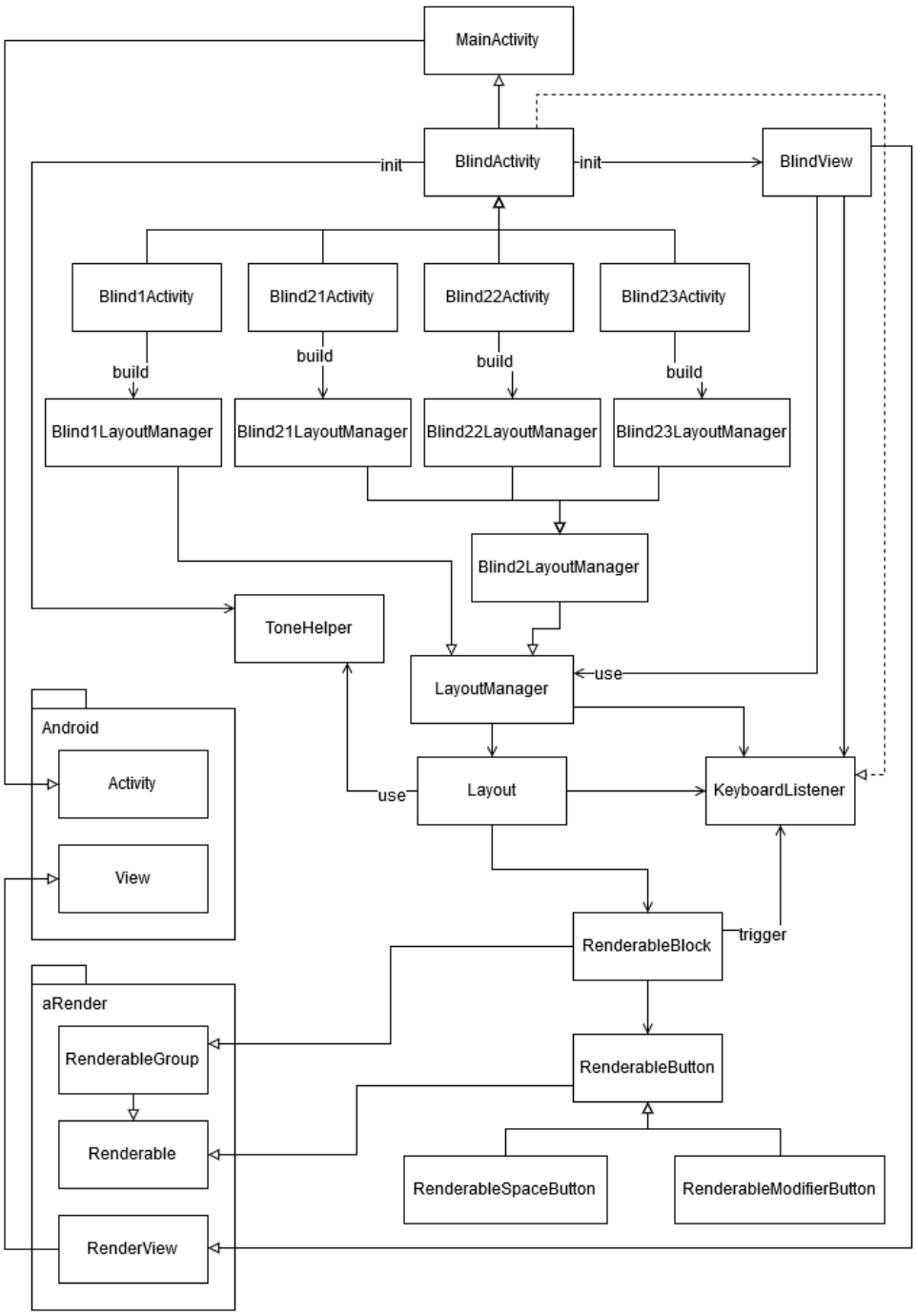

Figure 4.2.1: Overall architecture diagram.

Activity is an entry point to an Android application [23], so there are four Activities (one for each input method): *Blind1Activity*, *Blind21Activity*, *Blind22Activity*, and *Blind23Activity*. Since these Activities contain a lot of same functionality and corresponding code, the superclass *BlindActivity* is created, and common parts are moved to this superclass. Since the input method Simple-Touch can be useful for sighted people as well, parts that are not related to using by blind people are displaced to the superclass *MainActivity*. This solution simplifies creating a keyboard for sighted users in the future. Therefore: *MainActivity* contains common parts that are not related to using by blind people (for example, setting up a graphical library); *BlindActivity* contains parts that are related to using by blind people (for example, initialization of the object for audio feedback – *ToneHelper*); rest of Activities are lightweight, they are responsible for building objects (instances of subclasses of *LayoutManager*) according to the Factory Method design pattern [24]. *Blind1Activity* builds *Blind1LayoutManager*, which is responsible for the logic that is specific for the input method Simple-Touch; *Blind21Activity* builds *Blind21LayoutManager* (One-Touch); *Blind22Activity* builds *Blind22LayoutManager* (Two-Touches); *Blind23Activity* builds *Blind23LayoutManager* (Vary-Touches). Since last three input methods are similar, *Blind2LayoutManager* is introduced as a base class for their *LayoutManagers* to avoid code duplication.

LayoutManager (objects of subclasses of it) processes touch events to generate events for *Layout*, initializes keyboard blocks and selects *Layout*. In the current version of the input methods, just one *Layout* (alphabetical one) exists, but the application can be extended by additional layouts to support numeric and navigation buttons.

Layout is responsible for adjusting audio feedback via *ToneHelper* based on the position of blocks, sending events to *KeyboardListener*, building a graphical user interface using the aRender library.

ToneHelper holds logic of audio feedback and provides a simple interface for controlling that feedback (i.e. play, stop and set the volume for each channel individually).

BlindView is responsible for UI. It renders the graphical interface of the keyboard from *LayoutManager* and sends touch events from a system to *LayoutManager*.

KeyboardListener is an interface for events callback, which contains methods like *onCharacterTyped()* and *onBackspace()*. An Activity implements this interface and reacts to these events. Events can be raised by *LayoutManager*, *Layout* or *RenderableButton*.

aRender library is used for rendering graphic. This library is developed by the author of this thesis project. It uses the same way of rendering as standard Android components (*Views*) but provides a large set of utilities,

helpers, and classes for drawing common objects like text, bitmaps, groups, etc. That is why aRender is selected as the graphical library.

Core classes of aRender are *RenderView* and *Renderable*. *RenderView* is based on the system class *View*, which is the base class for widgets (so any UI made with aRender behaves like an Android component), and renders given *Renderable*. *Renderable* holds logic of rendering and measuring width and height of a UI component.

RenderableGroup is a *Renderable* that contains other *Renderables*; it is responsible for their positioning and rendering. *RenderableGroup* is developed according to the Composite design pattern [24].

RenderableBlock is a *RenderableGroup* that contains and renders *RenderableButtons*. *RenderableButton* holds UI and UX logic for buttons in general. It is extended by *RenderableSpaceButton* and *RenderableModifierButton* to support specific buttons.

4.3 Algorithms

This section describes the core algorithm of the adaptation logic – setting position, scale and rotation of a block relative to two touch points.

A keyboard block adapts to the position of a hand. As already stated, the adaptation means the transformation of the block to make two buttons of it to be exactly below two selected fingers. In other words, two points of the block (centers of related buttons) should have the same position as two touch points. Accent point (auxiliary term) is a point that has same coordinates as a point of a block if the block has 1:1 scale, 0 degrees tilt angle and is located at the point (0,0). Therefore, the accent point is not changed even after transformation of the block. Schematic image of the problem is shown in Figure 4.3.1. Accent points are white circles in the “original block” rectangle.

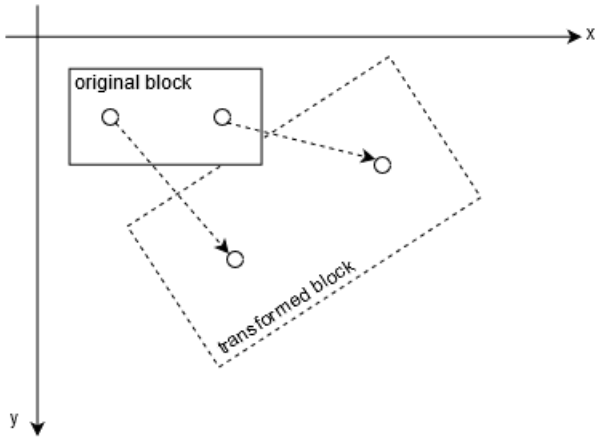

Figure 4.3.1: Schematic image of setting position, scale and rotation of a block relative to two touch points.

The problem can be described as a mathematical model that has 2 accent points and 2 touch points as input, and position, scale and tilt angle of the block as output. The problem-solving algorithm is developed and implemented using Java-like pseudocode. It is shown in Figure 4.3.2.

```
// input
PointF accentPoint0, accentPoint1;
PointF touchPoint0, touchPoint1;

// output
RenderableButtonsBlock block;

// scale
double handWidth = touchPoint1.x - touchPoint0.x;
double viewWidth = accentPoint1.x - accentPoint0.x;
if (viewWidth == 0) viewWidth = 1;
block.setScale(handWidth / viewWidth);

// position
block.setX(touchPoint0.x - block.getScale() * accentPoint0.x);
block.setY(touchPoint0.y - block.getScale() * accentPoint0.y);

// degree
double leftHandDX = touchPoint1.x - touchPoint0.x;
double leftHandDY = touchPoint1.y - touchPoint0.y;
double handDegree = Math.atan(leftHandDY / leftHandDX);
block.setRotation(handDegree);
block.setRotationPoint(accentPoint0.x, accentPoint0.y);
```

Figure 4.3.2: Java-like pseudocode for setting position, scale and rotation of a block relative to two touch points.

4.4 Implementation

The application is developed based on the described architecture and algorithm. The application implements normative description of each input method according to the described scope and limitations for ASUS TF700T. Screenshot of the application is shown in Figure 4.4.1.

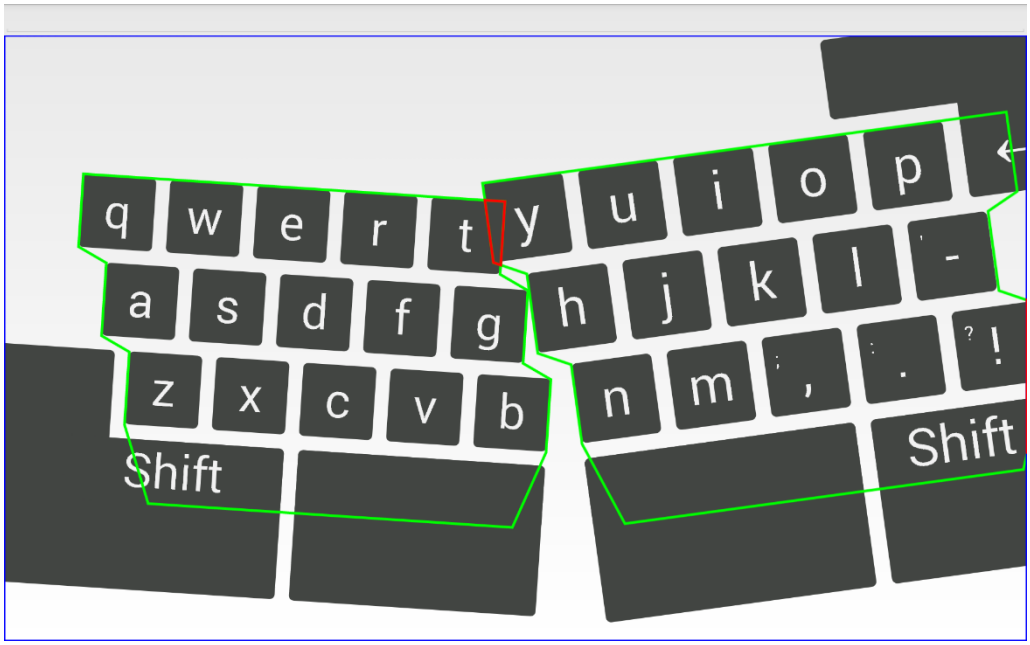

Figure 4.4.1: Screenshot of the developed application.

Moreover, additional features are implemented in the application to make it more useful for evaluation, experiments, and real-life usage:

- Areas of blocks are highlighted in green color, edges of the touch area – by blue color. Lines of overlapping of these areas are highlighted in red color. This feature allows detecting borders of areas and if areas are intercepted or not. This information is useful for a developer during development and debugging, and a supervisor during experiments.
- Typed text is displayed in a line at the top of the screen. It allows observing typed characters.
- The input method Simple-Touch can be initialized using not just 10 touches, but using 8 or 9 touches as well. It is implemented without significant changes to the input method because the input method does not use 2 lowermost touches.
- The input method Simple-Touch has an additional way to invoke the “Backspace” action – using gesture over any of space buttons. A user has to slide left over a “Space” button.

5 Evaluation

This chapter describes the evaluation of the developed input methods.

5.1 Method

This section describes an approach that is used to research the developed input methods. The developed input methods are compared with each other and with a standard accessible input method. It gave descriptive knowledge about advantages and disadvantaged of the developed input methods.

5.1.1 Approach

Inductive approach is selected because research papers do not contain enough information about input methods for touch panels that use the muscle memory from using a classic physical keyboard by blind people. Therefore, the research question is stated in Section 1.4 to be answered as a result of analysis of experiments.

Qualitative approach and observations are selected for input methods research, because the research requires a deep analysis, including WPM and error rate measurement during several sessions. It allows comparing the input methods with each other and understanding the learning effect of the input methods. Most research papers use same approach for input methods evaluation [1], [9], [10], [13], [14], [12].

The developed input methods are compared with an existing input method – TalkBack by Google. It is the default input method for blind people for Android devices (including ASUS TF700T).

5.1.2 Subjects selection

The main idea of the research is related to using the muscle memory. Users with more developed muscle memory from typing show the difference between the developed input methods and the existing input method better. Users, which use touch-typing, have more developed muscle memory than ones, which use 2-fingers typing. Therefore, users with the touch-typing skill are selected for experiments.

Users, which hold fingers on the home row keys between strokes, and users, which hold fingers in the air above the home row keys between strokes, can differ in the supreme input method as a result of the experiment. Therefore, the selection should contain people of different touch-typing styles.

Users with the skill of using TalkBack are not participating in the experiment since it could affect WPM and error rate of using TalkBack comparing with the developed input methods. Moreover, it could have an impact on the measurement of learning rate of TalkBack.

5.1.3 Procedure

All experiments are performed using the same device (ASUS TF700T), so all hardware-specific issues are the same for all input methods.

Fixed sentences are used instead of a phrases generator because using different phrases could affect results in a various and hard-predictable way. Three sentences from James & Reischel (2001) [25] are used for the experiments:

- hi joe how are you want to meet tonight
- want to go to the movies with sue and me
- what show do you want to see

Subjects are asked to type "as quickly and accurately as possible" [9]. MSD uncorrected error rate is measured instead of corrected error rate because corrected error rate is already measured as part of WPM and it is a time-consuming task to correct an error. The problem with this approach is that users have a different understanding of ”as accurately”; for example, some users want to correct any typo, but some users do not correct simple typos at all. This approach shows ”real-life usage” error rate.

The protocol of the experiment is as follow:

1. Ask a subject to practice with the input method “TalkBack” until the subject felt comfortable with it [14]. Users have different learning skills, so results of this step are not measured. Time for the adaptation is limited to 5 minutes for TalkBack, 5 minutes for the input method Simple-Touch, 3 minutes for the input method One-Touch and 2 minutes for rest of input methods. Time for the adaptation is not same, because input methods One-Touch, Two-Touches, and Vary-Touches are very similar to each other and similar to the input method Simple-Touch.
2. Ask the subject to look at the screen, which is located higher than subject’s eyes. The first sentence is displayed on the screen.
3. Ask the subject to type the sentence with the tablet, which is located on a table. The subject is not allowed to look at the tablet’s screen during this step.
4. Allow the subject having a rest after typing the sentence.
5. Repeat steps 2-4 with second and third sentences.
6. Repeat steps 1-5 with input methods Simple-Touch, One-Touch, Two-Touches and Vary-Touches.

Time of typing each sentence is measured instead of the total time of typing five sentences so that users could have a rest between typing sentences. Users are asked to type 5 sentences in a row with the same input method to measure learning effect.

The application makes a report that can be represented as a table or a sheet. Columns of the report table: input method name, time of action, and action. *Action* is the text ”onPositionUpdate” if input method blocks were changed (not related to TalkBack) or current text of the text field. Therefore, the application makes a new row in the report each time the user modifies the text of the text field (types something or uses ”Backspace”) or initializes the blocks. Time of any action is represented in milliseconds from midnight, January 1, 1970 UTC [26].

The report allows calculating WPM, MSD error rate and time for initialization. Since the report represents raw data, additional useful data can be extracted later on.

All the experiments were observed to get additional knowledge like:

- What are common mistakes and typos?
- Which problems do subjects have with initialization and typing?

The report gives knowledge “a typo has occurred”, but an observation gives knowledge “why the typo has occurred”. These questions and observation accent points are not fixed because probable mistakes could not be defined precisely enough before the experiment. Generally, the observer is matching any finger touch to desired button position, because it is the most common reason for mistakes.

All subjects were asked about touch-typing style (“Do you keep fingers lying on the home row between strokes or you hold fingers in the air above?”). The observer was also checking that a subject was not trying to cheat and look at the tablet’s screen.

For reliability purpose, subjects was not personally interested in success or fail of the project. The experiments was watched by the observer to avoid any cheating. If a subject cheated once – he (or she) was excluded from the experiment.

All verbal output from the subjects is briefly noted and processed to be included in Section 5.2. For ethical considerations, the output from the subjects (verbal one and typing results) is anonymized; the subjects got personal numbers to be referenced in the project. The subjects are not referred as “he” or “she” to avoid identification by a gender

During analysis, reports, which have been made by the application, are converted to the sheet with fields for an input method analysis, like WPM and error rate. Graphs and diagrams are made of this sheet. The observed behavior of using input methods and verbal feedback from subjects are searched for patterns.

5.2 Results/Analysis

The experiments are performed on four subjects. They are called Subject 1, …, Subject 4 to keep results anonymous. The results of the experiments are

divided into two subsections: formal and empirical ones. The formal results are about reports generated by the application and any data derived directly from them. The empirical data contains results of observation and verbal output from the subjects.

5.2.1 Formal results

Report files are generated by prototype applications and imported into a spreadsheet to simplify further processing. A sample of such raw results is represented in Table 5.2.1.

Action #	Input method name	Time of action, milliseconds	Action
1	Blind1Activity	0	onPositionUpdate
2	Blind1Activity	114	onPositionUpdate
…			
29	Blind1Activity	2958	onPositionUpdate
30	Blind1Activity	4355	h
31	Blind1Activity	4906	hi
32	Blind1Activity	5525	hi
33	Blind1Activity	6472	hi j
34	Blind1Activity	7082	hi jp
35	Blind1Activity	8679	hi j
36	Blind1Activity	9670	hi jp
37	Blind1Activity	10661	hi j
38	Blind1Activity	11403	hi jo
39	Blind1Activity	12486	hi joe
…			
91	Blind1Activity	65880	hi joe how aree you want to meet tonu ight

Table 5.2.1: A sample of a raw report that is generated by a prototype application and imported into a spreadsheet.

The algorithm that is used to extract data from raw reports:

1. Recognize the resulting sentence in the list of actions and find the first action that happened with the final text. It is the message “hi joe how aree you want to meet tonu ight”, action #91.
2. Recognize the action that started the latest initialization before the first action of typing the sentence. It is action #1.
3. Use the following formula to calculate WPM:

$$\mathrm{WPM} = \frac{(\mathrm{W} - 1)/5}{\mathrm{M}}$$

where: W is a number of characters (including spaces), M is a time in minutes [3].
Since all measurements are represented in milliseconds, formula is transformed to:

$$\mathrm{WPM} = \frac{((\mathrm{W}-1)*60*1000)/5}{\mathrm{S}} = \frac{(\mathrm{W}-1)*12000}{\mathrm{S}}$$

where: S is a time in milliseconds.
S equals the time between the action from the first step and the action from the second step, so S=65880-0=65880. The desired text is "hi joe how are you want to meet tonight", so W=39. Therefore:

$$\mathrm{WPM} = \frac{(39-1)*12000}{65880} = 6.92\ (\text{approx.})$$

4. Calculate MSDER (MSD error rate) using the following formula:

$$\mathrm{MSDER} = \frac{\mathrm{MSD}(\mathrm{P},\mathrm{T})}{\mathrm{MAX}(|\mathrm{P}|,|\mathrm{T}|)}$$

where: P is an original text, T is a typed text [3].
Levenshtein function is used for MSD calculation. MSD equals 3 for this case. Therefore, MSDER = 3/42 = 0.07 (aprox.)

Therefore, the output is WPM and MSD error rate; it is suitable for comparison with existing input methods like TalkBack. Sometimes the subjects were warming-up fingers between typing of sentences. It leads to typing random characters. These situations are considered manually, and they do not affect WPM and error rate.

The subjects were supposed to have exactly one initialization before each sentence, but some of them were doing the initialization for a few times in a row or just do not do reinitialization between sentences at all. Therefore, the initialization affects WPM of typing different sentences in a various way, so comparison WPM of typing sentences of the same subject and input method between each other is not reliable. Therefore, the measurement of learning effect does not give a reliable result for these experiments. However, the input methods can still be compared by calculating average WPM and error rate across all three sentences. The learning effect can be investigated in a further research of the results by splitting typing time to initialization time and time of typing itself. In this case, the learning effect can be measured for initialization and typing independently in order to find bottlenecks ("Is the bottleneck related

to initialization or typing?"). Time of hand lifting (the time between the last action of initialization and first typed character) should be ignored, because it does not depend on initialization or typing process.

For example, the calculation of the time for initialization of blocks *t* and *WPM* of typing itself for sample from Table 5.2.1:

$$t = 2958 - 0 = 2958 \text{ (milliseconds)}$$

$$WPM = \frac{(39-1) * 12000}{65880 - 4355} = 7.41 \text{ (aprox.)}$$

Raw results are converted into a sheet. Each row of it represents information on typing one sentence by specific subject using specific input method. Sheet has following columns:

- "Subject": a number of a subject.
- "Touch-typing style": represents subject's answer to the question "Do you keep fingers lying on the home row between strokes or you hold fingers in the air above?". The column takes one of three values: "keep fingers on the home row", "keep fingers in the air", "mixed" (the behavior is different depending on the situation). The answer of each subject is checked by observing subject's way of using a physical keyboard.
- "Input method": which input method is used to type a sentence.
- "Start time": time of start of initialization before typing the sentence (or just time of start typing for the case of TalkBack).
- "End time": time of typing the last character of the sentence.
- "Time": time of typing the sentence including time of the initialization for non-TalkBack cases. This column is calculated automatically by the spreadsheet as the difference between "Start time" and "End time".
- "Typed sentence": a sequence of characters that the user has typed.
- "Target sentence": a sequence of characters that the user was asked to type.
- "Target sentence length": length of "Target sentence".
- "WPM": words per minute. Calculated automatically by the spreadsheet using the described algorithm and columns "Time" and "Target sentence length".
- "MSD": Levenshtein distance between "Typed sentence" and "Target sentence", calculated automatically by the spreadsheet.
- "MSDER": MSD error rate; calculated automatically by the spreadsheet using the described algorithm and columns "MSD", the length of "Typed sentence" and "Target sentence".

The perfect number of rows (number of experiments) would be S*I*T, where S is a number of the subjects, I is a number of the input methods, T is a number of the sentences. Therefore, the expected number of rows was 4*5*3=60, but some of the experiments were canceled or rejected.

Subject 1 was too stressed during typing the first sentence using the developed input method One-Touch, so further experiments with this subject were cancelled. Subject 1 pointed the following reason of the stress: "The input method One-Touch requires too unaccustomed movements to type characters".

Experiments with second and third sentences were cancelled after an experiment with the first sentence because of too high error rate (over 0.4) for following cases: subject 2, the input method One-Touch; subject 2, the input method Two-Touches; subject 2, the input method Vary-Touches; subject 4, the input method Two-Touches. Following experiments were going to be cancelled because of the same reason, but subjects asked to continue with it because they believed that they can type last two sentences better: subject 2, the input method "TalkBack"; subject 4, the input method Vary-Touches. The assumption was valid for both cases (the subjects have shown lower error rate for last two sentences). The resulting sheet is provided in Appendix A.

Mean and standard deviation of WPM and MSD error rate (MSDER) are calculated automatically using the spreadsheet. These values calculated for each sentence (Table 5.2.2), and input method in general (Table 5.2.3).

Input method	Sentence	WPM mean	WPM standard deviation	MSDER mean	MSDER standard deviation
TalkBack	1	3,77	2,01	0,19	0,24
TalkBack	2	3,59	1,19	0,06	0,07
TalkBack	3	3,41	1,19	0,02	0,02
Simple-Touch	1	6,00	2,60	0,13	0,06
Simple-Touch	2	4,53	2,29	0,12	0,10
Simple-Touch	3	5,81	1,41	0,06	0,06
One-Touch	1	3,84	0,14	0,47	0,33
One-Touch	2	5,71	-	0,40	-
One-Touch	3	4,31	-	0,43	-
Two-Touches	1	3,39	0,90	0,43	0,03
Two-Touches	2	-	-	-	-
Two-Touches	3	-	-	-	-
Vary-Touches	1	6,46	4,31	0,52	0,06
Vary-Touches	2	8,77	-	0,31	-
Vary-Touches	3	7,56	-	0,19	-

Table 5.2.2: Performance of typing per each sentence.

Input method	WPM mean	WPM standard deviation	MSDER mean	MSDER standard deviation
TalkBack	3,59	1,38	0,09	0,15
Simple-Touch	5,45	2,07	0,10	0,08
One-Touch	4,43	0,89	0,45	0,19
Two-Touches	3,39	0,90	0,43	0,03
Vary-Touches	7,31	2,72	0,39	0,17

Table 5.2.3: Performance of typing per each input method.

As already mentioned, the learning effect is not reliable for these measurements of the developed input methods because of initialization time, but the graph of Figure 5.2.1 shows that three sentences are not enough to measure the learning effect, even for TalkBack, because the WPM is decreasing that is not expected.

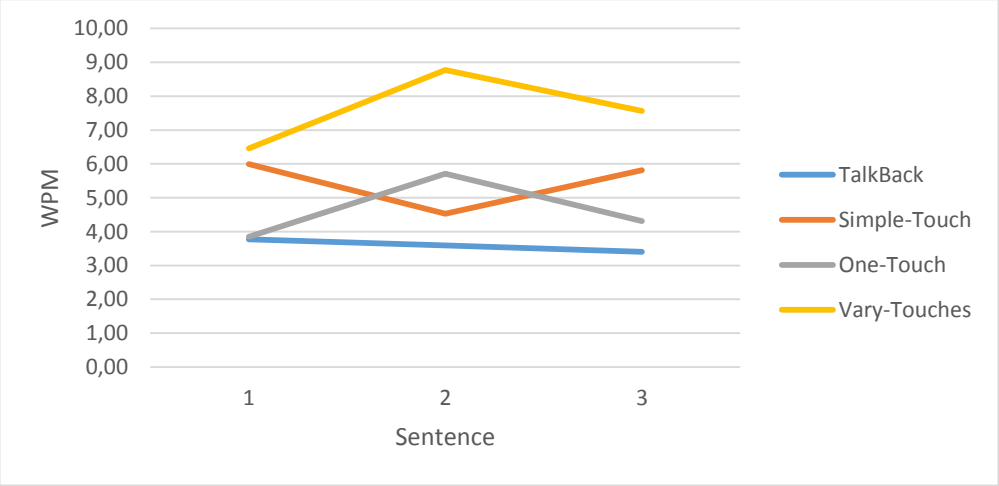

Figure 5.2.1: WPM over sentences.

Additionally, the experiment is performed on the developer (subject 5) of the input methods and the prototypes, because this person was practicing a lot with the input methods during the development. The experiment was simplified by excluding TalkBack, because the person has spent much less time with TalkBack than with the developed input methods, so the comparison would be unfair. Since the time of learning was not measured and it is scattered across around one year, WPM and error rate are not used for calculating average WPM and error rate of subjects 1-4 (experiments with the subject 5 do

not affect Table 5.2.2, Table 5.2.3 and Figure 5.2.1). Important to mention that the person has just the basic skill of touch-typing, so the picture of the standard QWERTY keyboard layout was shown to the person during the experiment. The goal of this experiment is to roughly measure possible learning effect of the developed input methods. Performance (WPM and error rate) of each input method is visualized in Figure 5.2.2 and Figure 5.2.3.

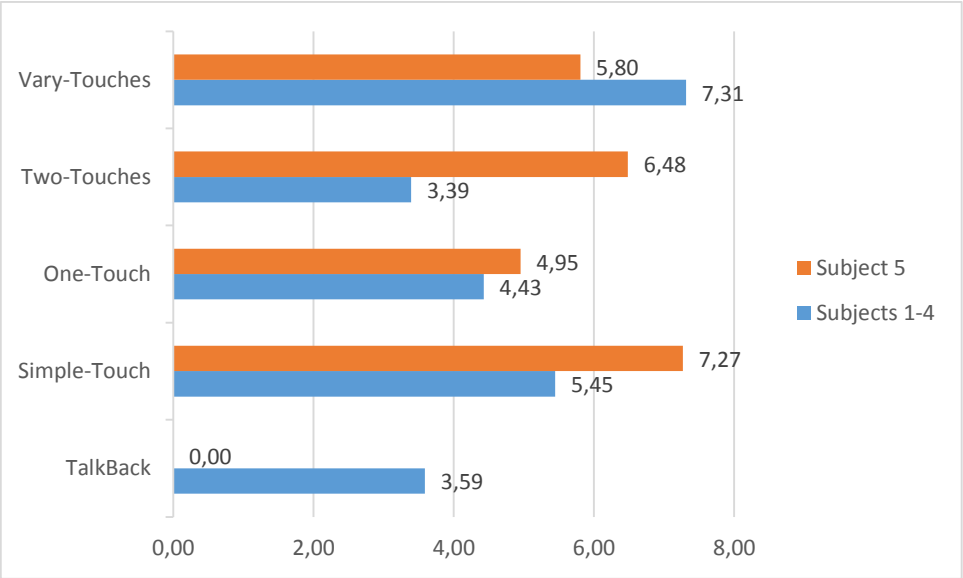

Figure 5.2.2: WPM over input methods.

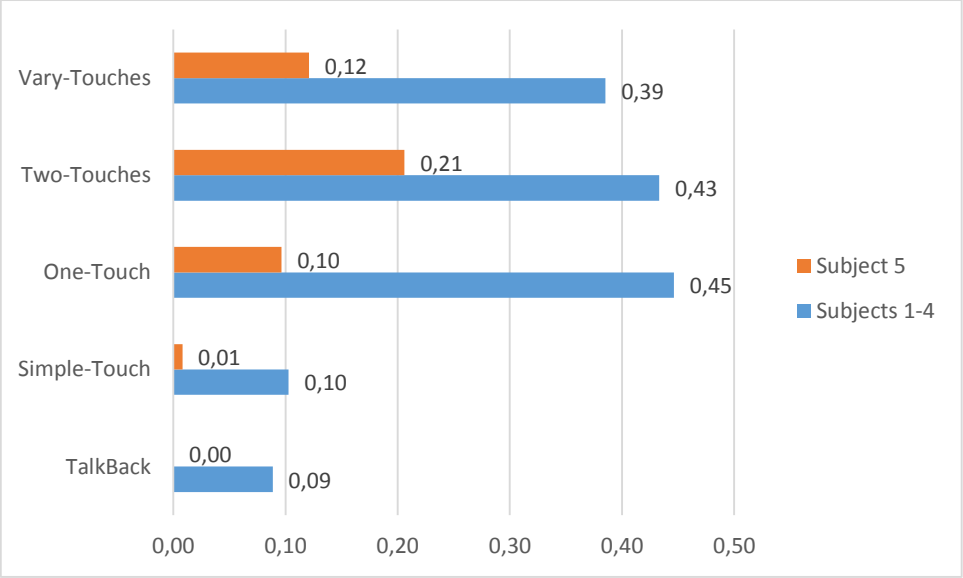

Figure 5.2.3: MSD error rate over input methods.

Generally, the performance of the developed input method Simple-Touch is better than the performance of the standard input method, but input methods One-Touch, Two-Touches, and Vary-Touches give much higher error rate than the standard input method.

5.2.2 Empirical results

Each subject was participating an experiment for about 50 minutes, including time for education, evaluation of the input methods and giving verbal feedback. All subjects agreed that the developed input method Simple-Touch is less stressful than input methods One-Touch, Two-Touches, and Vary-Touches because the typing style of it is most similar to a standard physical keyboard.

During typing the second sentence using TalkBack, subject 1 mentioned that it would be very hard to type using TalkBack without the skill of touch-typing skill because the direction of a finger movement is determined by the knowledge of the keyboard layout.

After experiments, subject 2 has mentioned that it would be useful to feel a vibration when notched keys ("F" and "J") are touched and have audio feedback on a key touch before the character is typed (same as TalkBack).

The followings were discovered during observation of experiments with 1-4 subjects:

1. All subjects do not use to hold thumbs on space button, neither during initialization nor typing. Therefore, they were intuitively lifting the thumbs. It leads to mistakes, in particular for input methods One-Touch, Two-Touches, and Vary-Touches because they require holding the thumbs on the space button for most of the time. Similar problem with the position of the thumbs: all subjects were placing the thumbs above the space button. The problem is not relevant for the input method Simple-Touch because this input method does not require thumbs touches.
2. Subject 2 has tried to type "H" key right after "Y" key without returning to the base position during typing with the input method Two-Touches because the subject uses the same finger to type these two buttons and he does same for physical keyboards. However, it led to a mistake, because the input method requires the fingers to be returned to the base position before typing next character.
3. During experiments with input methods One-Touch, Two-Touches, and Vary-Touches, a touch panel was not receiving all required touches sometimes. The most common reasons for it: subjects lift up fingers instinctively (especially thumbs, as mentioned in the first recognized pattern); the touch panel does not recognize touches because of lack of sensitivity (especially thumbs touches, because contact area of a thumb and the touch panel is small in the case of the natural position of wrists).

4. All subjects had problems with switching between input methods One-Touch, Two-Touches, and Vary-Touches because each variant requires different movements for typing character. Subjects were trying to use movements of previous input method for first words of a sentence, but this effect becomes weaker with each typed character.

5.3 Discussion

This section discusses results and the implementation of the study. Main sources of knowledge are Figure 5.2.2 and Figure 5.2.3. The developed input method Simple-Touch gives better results than TalkBack even during typing three sentences. It is reflected on WPM, error rate and feedback by subjects about the level of stress of using the input method. Moreover, all four recognized patterns affect the experience of using an input method in a negative way, and none of them is about the input method Simple-Touch. An additional reason for the input method success is that it does not require the panel to be touched by thumbs.

5.3.1 Input methods One-Touch, Two-Touches, Vary-Touches

Three sentences are not enough to make a subject learn an input method to keep error rate lower than 0.3 for input methods One-Touch, Two-Touches, and Vary-Touches. An input method with error rate that is higher than 0.3 is barely usable for everyday usage, but the experiment with the subject 5 has shown that it is possible to lower error rate to 0.21, so these input methods can be evaluated more detailed with large quantity of sentences to prove that the average user could lower error rate to the satisfactory level.

Priority in the research should be given to the input method Vary-Touches because it has shown generally better WPM and error rate. The need to decrease the number of researched input methods is also confirmed by the fourth recognized pattern.

5.3.2 Method reflection

Subjects need more than three sentences to start getting used to an input method. It is proven by Figure 5.2.1 and the fourth recognized pattern. Investigation of the learning effect of the developed input methods can be improved by splitting typing time to initialization time and time of typing itself. In this case, the learning effect can be measured independently for initialization and typing in order to find bottlenecks.

In the performed research WPM has been calculated with taking into account time for initialization, but this time is directly proportional to the size of the touch panel. Therefore, the same experiment could give a greater superiority of the developed input methods over TalkBack for the case of wider touch screens.

5.3.3 Comparison the developed input methods with existing ones

Comparison the developed input method with existing ones requires additional experiments with the existing input methods, because all experiments should be performed in the same conditions (same subjects, same texts, same devices, etc). For example, long sentences with complex words are most likely typed with lower WPM than short sentences with simple words. Therefore, it is not correct just to compare WPM and error rate of the developed input methods with the same metrics of the existing methods that are measured under other conditions in terms of experiments of another research papers.

WPM and error rate of the developed input methods are compared with the input methods that are evaluated in the paper "Survey of Eye-Free Text Entry Techniques of Touch Screen Mobile Devices Designed for Visually Impaired Users" [1]. Results of the comparison are represented in Table 5.3.1, Figure 5.3.1 and Figure 5.3.2 to give a rough understanding of the developed input methods relatively to existing ones.

Input method	WPM	MSD error rate, %
TalkBack	3,59	9,00
Simple-Touch	5,45	10,00
One-Touch	4,43	45,00
Two-Touches	3,39	43,00
Vary-Touches	7,31	39,00
MultiTap	0,88	15,28
SVIFT	4,75	-
Unisroke	32	-
Graffiti	7,6	0,40
Slide Rule	27	14,10
VoiceOver	0,66	9,70
QWERTY	2,11	5,20
BrailleType	1,45	9,70

Table 5.3.1: Performance of typing of the developed input methods and the existing input methods.

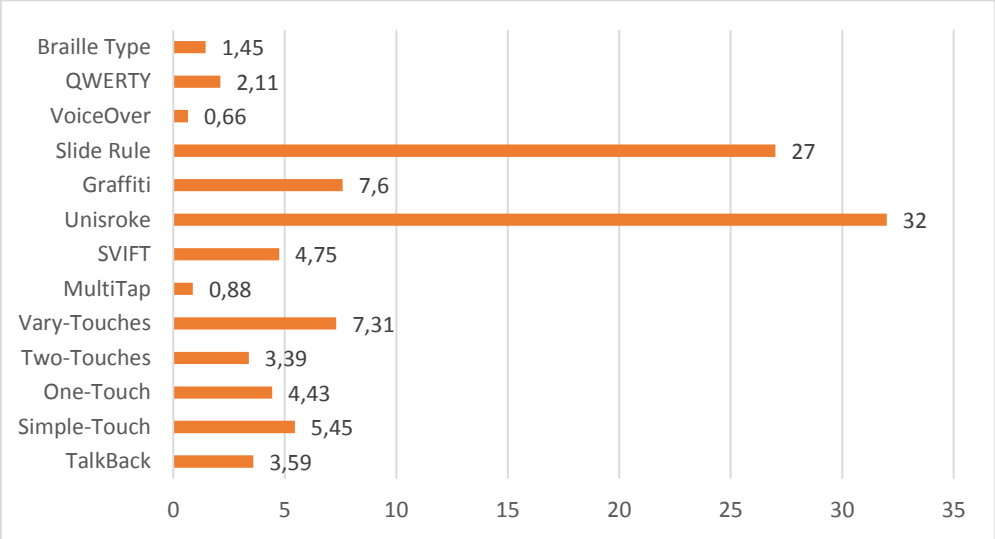

Figure 5.3.1: WPM over the input methods of Table 5.3.1.

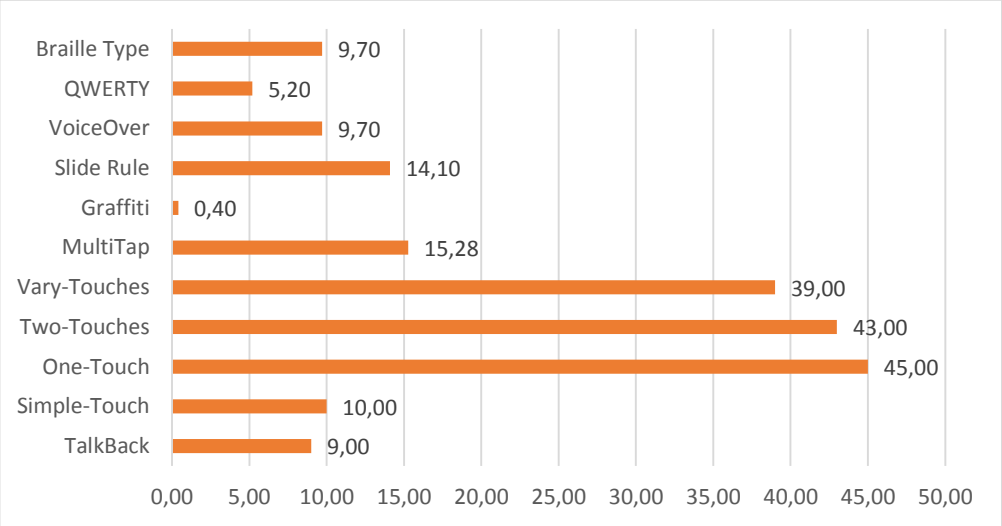

Figure 5.3.2: MSD error rate over the input methods of Table 5.3.1.

6 Conclusions

In this project, existing input methods for blind people were analyzed to find a method to improve the typing performance of blind people on wide touch panels. One solution to this problem is to take the advantage of muscle memory from using classic physical keyboards. To this end, four input methods Simple-Touch, One-Touch, Two-Touches, and Vary-Touch are designed, and a prototype for each one is developed. These input methods are compared with each other and with a standard input method.

The project has shown that performance of typing by blind people on wide touch panels can be improved by using the muscle memory from using physical QWERTY keyboards. The experiment showed the effectiveness of each approach and discussed their advantages and disadvantages. The developed input method Simple-Touch has a better WPM than TalkBack and has the same error rate. The developed input method Vary-Touch gives critically high error rate, but its WPM is promising. Input methods One-Touch and Two-Touches give higher error rate and lower WPM than Vary-Touch.

For further evaluation of the input methods developed in this project, extensive experiments with more subjects are planned to be carried out in the future. Since input methods One-Touch and Two-Touches did not perform well, subjects would have more time to participate experiments with the rest of the input methods, so it makes sense to repeat the experiment for just three input methods (TalkBack, Simple-Touch, and Vary-Touches). The following modifications will be made in the future experiments:

- Use more sentences to improve the visibility of learning effect.
- Measure time of initialization and time of typing itself independently.
- Make some experiments with wider touch panels to estimate the dependence of input method performance on the width of the touch panel.
- Improve the input method Vary-Touches according to the recognized patterns.

As another future work, one can modify the input method Simple-Touch for use by sighted people to increase their typing speed comparing to a standard touch input method for sighted users.

The findings of the project show that the performance of typing is decreased due to the low sensitivity of touch panels. Software that will recognize the pattern of touch actions in runtime to recover missed touches will be used to solve this problem partially.

References

[1] S. Banubakode and C. Dhawale, "Survey of Eye-Free Text Entry Techniques of Touch Screen Mobile Devices Designed for Visually Impaired Users," *Covenant Journal of Informatics and Communication Technology (CJICT) Vol*, vol. 1, 2013.

[2] Evengrounds.com. (2013) Do Blind People Use a Special Keyboard?. [Online]. Available: http://evengrounds.com/blog/do-blind-people-use-special-keyboard

[3] I. S. MacKenzie and K. Tanaka-Ishii, *Text entry systems: Mobility, accessibility, universality*: Morgan Kaufmann, 2010.

[4] Wiktionary. (2016) muscle memory. [Online] Available: https://en.wiktionary.org/wiki/muscle_memory

[5] Google Inc. (2016) Creating an Input Method | Android Developers. [Online] Available: http://developer.android.com/intl/ru/guide/topics/text/creating-input-method.html

[6] Besttyping.com. (2015) What is Touch Typing_Touch Typing. [Online] Available: http://www.besttyping.com/

[7] S. P. Mariotti, "Global data on visual impairments 2010," *World Health Organization*, vol. 20, 2012.

[8] T. R. Ostrach, "Typing speed: How fast is average: 4,000 typing scores statistically analyzed and interpreted," Orlando, FL: Five Star Staffing, 1997.

[9] S. Azenkot, "Eyes-Free Input on Mobile Devices," University of Washington, 2014.

[10] J. Oliveira, T. Guerreiro, H. Nicolau, J. Jorge, and D. Gonçalves, "BrailleType: unleashing braille over touch screen mobile phones," in *Human-Computer Interaction–INTERACT 2011*, ed: Springer, 2011, pp. 100-107.

[11] Apple Inc. (2016) Accessibility - iOS - VoiceOver - Apple. [Online] Available: http://www.apple.com/accessibility/ios/voiceover/

[12] J. Guerreiro, A. Rodrigues, K. Montague, T. Guerreiro, H. Nicolau, and D. Gonçalves, "TabLETS Get Physical: Non-Visual Text Entry on Tablet

Devices," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 39-42.

[13] S. J. Castellucci and I. S. MacKenzie, "Graffiti vs. unistrokes: an empirical comparison," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008, pp. 305-308.

[14] L. Butts and A. Cockburn, "An evaluation of mobile phone text input methods," *Australian Computer Science Communications*, vol. 24, pp. 55-59, 2002.

[15] M. Silfverberg, I. S. MacKenzie, and P. Korhonen, "Predicting text entry speed on mobile phones," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, 2000, pp. 9-16.

[16] W. C. Westerman, "System and method for recognizing touch typing under limited tactile feedback conditions," ed: Google Patents, 2004.

[17] S. Sameer, *Complete Computer Hardware Only*, PediaPress, 2011.

[18] M. Shetter, "Virtual keyboard for touch-typing using audio feedback," ed: Google Patents, 2005.

[19] D. Rempel, A. Barr, D. Brafman, and E. Young, "The effect of six keyboard designs on wrist and forearm postures," *Applied ergonomics*, vol. 38, pp. 293-298, 2007.

[20] Crunchbase.com. (2015) Dryft | CrunchBase. [Online] Available: https://www.crunchbase.com/organization/dryft

[21] Apple Inc. (2015) Use third-party keyboards on your iPhone, iPad, and iPod touch - Apple Support. [Online] Available: https://support.apple.com/en-us/HT204340

[22] Apple Inc. (2016) iPad Pro - Apple. [Online todo] Available: http://www.apple.com/se/ipad-pro/

[23] Google Inc. (2016) Activities | Android Developers. [Online] Available: http://developer.android.com/guide/components/activities.html

[24] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software: Pearson Education, 1994.

[25] C. L. James and K. M. Reischel, "Text input for mobile devices: comparing model prediction to actual performance," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2001, pp. 365-371.

[26] Oracle. (2016) System (Java Platform SE 7). [Online] Available: https://docs.oracle.com/javase/7/docs/api/java/lang/System.html#currentTimeMillis%28%29

A Appendix "Results sheet"

The results sheet is simplified by excluding columns "Start time", "End time", "Target sentence length" and "MSD" to decrease the size of it.

Sub-ject	Touch-typing style	Input method	Time	Typed sentence	Target sentence	WPM	MSDER
1	mixed	TalkBack	167407	hi joe how are uou want to meet tionight	hi joe how are you want to meet tonight	2,723900434	0,05
1	mixed	TalkBack	130433	want to go to the movies with sue and me	want to go to the movies with sue and me	3,588049037	0
1	mixed	TalkBack	143709	what show do you ant to see	what show do you want to see	2,254556082	0,035714286
1	mixed	Simple-Touch	65880	hi joe how aree you want to meet tonu ight	hi joe how are you want to meet tonight	6,921675774	0,071428571
1	mixed	Simple-Touch	100817	wamt to g to the mof vies e with sue a d me	want to go to the movies with sue and me	4,642074253	0,181818182
1	mixed	Simple-Touch	65583	what shoe do uou wz ant to see	what show do you want to see	4,940304652	0,133333333
2	mixed	TalkBack	76197	hijoehowareyouwant	hi joe how are you want to meet tonight	5,984487578	0,538461538
2	mixed	TalkBack	125623	want to go to the mobies with sue and me	want to go to the movies with sue and me	3,725432445	0,025
2	mixed	TalkBack	96225	what show do you want to see	what show do you want to see	3,36710834	0
2	mixed	Simple-Touch	194841	hi joe how are uouwamrctto neet tomight	hi joe how are you want to meet tonight	2,34036984	0,205128205
2	mixed	Simple-Touch	142256	want to ho to the mobies with sue and me	want to go to the movies with sue and me	3,289843662	0,05

2	mixed	Simple-Touch	75245	what show do you want to see	what show do you want to see	4,305933949	0
2	mixed	One-Touch	121769	hijoe j-mhlow ffsadfgxxxxre u oi eamg fffo ,utf	hi joe how are you want to meet tonight	3,744795473	0,708333333
2	mixed	Two-Touches	165493	hi joe jjjj, how at hjoo wamt to mrwseet tfaaom,ihj	hi joe how are you want to meet tonight	2,755403552	0,454545455
2	mixed	Vary-Touch	133823	go jjoe jows aafrde uoi wsabfm fftoo mrbeef ffoon mmjmoiggft	hi joe how are you want to meet tonight	3,407486008	0,560606061
3	keep fingers on the home row	TalkBack	296732	hi joe how are you wantto mt tonight	hi joe how are you want to meet tonight	1,536740224	0,076923077
3	keep fingers on the home row	TalkBack	226022	want to tgoto the mobies ith sue and mn	want to go to the movies with sue and me	2,070594898	0,15
3	keep fingers on the home row	TalkBack	109675	what show do uou want to see	what show do you want to see	2,954182813	0,035714286
3	keep fingers on the home row	Simple-Touch	54004	ji joe how are you want tk neet tonighewt	hi joe how are you want to meet tonight	8,443818976	0,12195122
3	keep fingers on the home row	Simple-Touch	187218	wang to glo to to tnphrmiobies with sue and me	want to go to the movies with sue and me	2,499759638	0,239130435

Linnæus University
School of Computer Science, Physics and Mathematics

351 95 Växjö / 391 82 Kalmar
Tel 0772-28 80 00
dfm@lnu.se
Lnu.se

3	keep fingers on the home row	Simple-Touch	47219	what shjow do you want to see	what show do you want to see	6,861644677	0,034482759
4	keep fingers in the air	TalkBack	94444	hi joe how sare youwant tomeet tonight	hi joe how are you want to meet tonight	4,828258015	0,076923077
4	keep fingers in the air	TalkBack	93818	wantto go to the mmo ies with sue and me	want to go to the movies with sue and me	4,98838176	0,075
4	keep fingers in the air	TalkBack	64225	what show do you want to see	what show do you want to see	5,0447645	0
4	keep fingers in the air	Simple-Touch	72566	hi s how are oyou want to meet tonight	hi joe how are you want to meet tonight	6,283934625	0,102564103
4	keep fingers in the air	Simple-Touch	60839	want to go to the movies with sue and me	want to go to the movies with sue and me	7,692434129	0,024390244
4	keep fingers in the air	Simple-Touch	45305	what show do uofu want to see	what show do you want to see	7,151528529	0,068965517
4	keep fingers in the air	One-Touch	115643	hi joe how fFare uou want too .n meet tonomight	hi joe how are you want to meet tonight	3,943169928	0,24
4	keep fingers in the air	One-Touch	81971	wa.frrdml rdto ho the movies with sime amjkk,ke	want to go to the movies with sue and me	5,709336229	0,404255319
4	keep fingers in the air	One-Touch	75136	wantshow ssddimm uoi swant to to see	what show do you want to see	4,312180579	0,432432432
4	keep fingers in the air	Two-Touches	113109	jhi j-loe howare uok wammiitf c co mreset tomigjuu	hi joe how are you want to meet tonight	4,031509429	0,411764706

Linnæus University
School of Computer Science, Physics and Mathematics

351 95 Växjö / 391 82 Kalmar
Tel 0772-28 80 00
dfm@lnu.se
Lnu.se

4	keep fingers in the air	Vary-Touch	47969	hi jj olj jd hjofm fwaan f to meet tomigjjt	hi joe how are you want to meet tonight	9,506139382	0,479166667
4	keep fingers in the air	Vary-Touch	53362	waa jt to go to the movirs witjj siose aanf me	want to go to the movies with sue and me	8,770285971	0,31372549
4	keep fingers in the air	Vary-Touch	42842	what show do uoi wanyt to see	what show do you want to see	7,562672144	0,1875
5	keep fingers in the air	Simple-Touch	58467	hi joe how are you want to meet tonight	hi joe how are you want to meet tonight	7,799271384	0
5	keep fingers in the air	Simple-Touch	68696	want to go to the movies with sue and mme	want to go to the movies with sue and me	6,812623734	0,024390244
5	keep fingers in the air	Simple-Touch	45053	what show do you want to see	what show do you want to see	7,191529976	0
5	keep fingers in the air	One-Touch	66847	hi joe how are uou wamt meet tonighr	hi joe how are you want to meet tonight	6,821547713	0,153846154
5	keep fingers in the air	One-Touch	112567	want fo go the movies with sue and me	want to go to the movies with sue and me	4,157523964	0,1
5	keep fingers in the air	One-Touch	83525	what show do uou want to see	what show do you want to see	3,87907812	0,035714286
5	keep fingers in the air	Two-Touches	85151	hi joe hjl-ow are uou want me meet tonight	hi joe how are you want to meet tonight	5,35519254	0,142857143
5	keep fingers in the air	Two-Touches	96894	want to go the movies wupith sue amnd kddnd me	want to go to the movies with sue and me	4,830020435	0,260869565
5	keep fingers in the air	Two-Touches	34965	what show dou want see	what show do you want to see	9,266409266	0,214285714

Linnæus University
School of Computer Science, Physics and Mathematics

351 95 Växjö / 391 82 Kalmar
Tel 0772-28 80 00
dfm@lnu.se
Lnu.se

5	keep fingers in the air	Vary-Touch	81153	hi joe ho ared juojj want to meet tonigjjt	hi joe how are you want to meet tonight	5,619015933	0,227272727
5	keep fingers in the air	Vary-Touch	77270	want togo to tje movies with sue amd kme	want to go to the movies with sue and me	6,056684354	0,1
5	keep fingers in the air	Vary-Touch	56456	what show do uou want to see	what show do you want to see	5,73898257	0,035714286

Linnæus University
School of Computer Science, Physics and Mathematics

351 95 Växjö / 391 82 Kalmar
Tel 0772-28 80 00
dfm@lnu.se
Lnu.se