A new Technique for Text Entry on Small Mobile Devices - Summary

- SW10 master thesis by d612a -

- Group d612a -
Anders Houbak Kristiansen
Frederik Larsen

Aalborg University

The Department of Computer Science
Aalborg University

Software Engineering, 10th semester

Title:

A new Technique for Text Entry on Small Mobile Devices - Summary

Project period:
February 2nd 2009 - June 15th, 2009

Theme:
HCI Master Thesis

Group:
d612a

Authors:
Anders Houbak Kristiansen
Frederik Larsen

Supervisor:
Jan Stage

Number printed: 4

Pages: 33

Finished: June 15th, 2009

Abstract

This study consists of a two articles and a summary of those. They deal with the question: *How can we improve text entry on small touchscreen based mobile devices?*. To answer this we have developed a new technique for writing text on small touchscreen based mobile devices, called directional selection. This technique was then evaluated through the two articles, using two different perspectives.
The first article investigates this technique to determine if it is feasible. It deals with the question: *Can directional selection be used for text entry on small touchscreen based mobile devices?*.
The second article compares an implementation of the technique against several existing text entry techniques. It deals with the question: *How does QdQ compare to existing small soft keyboards?*.

Project group d612a

Anders Houbak Kristiansen	Frederik Larsen

Contents

1 Introduction

Small mobile devices are used in many new areas where the desktop or laptop computer used to dominate. Devices like smartphones are used for things like browsing homepages, writing emails and participate in social networks. A good example of this is that many different smartphones manufacturers like Apple, Blackberry, Sony Ericsson and many more have made special Facebook(social network) application for their smartphones. All of these new requirements require the smartphones to be able to write text with a decent efficiency in many different contexts.

When smartphones first came out, they used a hardware keyboard usually based on T9 by Tegic [1], which is a 12 key keyboard that uses a dictionary, or they used a very small QWERTY keyboard. This has changed with the introduction of new smartphones equipped with only a couple of buttons and a relatively large touchscreen covering almost the entire device like the iPhone from Apple and HTC Touch Diamond2. Some of the reasons to do this is the phones becomes small and easier to make since they do not need to have the physical keyboard and the mechanism to show and hide it.

Many of these new smartphones including the two named before, uses a capacitive touchscreen that works by using the small amount of electricity the human body conducts. This means they will not work with a stylus. There are many reasons to use this type of touchscreen, compared to resistive touchscreens that uses the pressure on the screen and which is found in many older devices. First the screen can have a glass like coat since it does not need to be flexible, this make the smartphone look less cheap and more aesthetically pleasing. Capacitive touchscreens also has a higher clarity then a resistive touchscreen. These reasons together with it is more durable will in terms make the mobile manufacturer sell more smartphones. This is somewhat of a problem because most of the researches done in the field of writing on soft keyboards (a keyboard on a touchscreen) have been based on using a stylus, which will not work on this type of touchscreen. There is also a good reason for not using a stylus, it is that the intense attention required can disrupt the users attention from the actual text input and it can feel tedious using a stylus over an extended period of time[6].

This master thesis focuses on a new finger based technique for typing text on small touchscreen based devices. One of the goals of this technique is to be independent of dictionaries, which has two important drawbacks we want to avoid. The first is, as [4] shows, that if only 15% of the words written is not in the dictionary all speed increase are lost. This can be words like abbreviations, usernames, passwords, email and web page address, which all are very important for the new areas the smartphones are used in. The other reason is the high cognitive load that the users feel when they are writing, because they need to focus on what the dictionary shows on the screen and on the next key they need to press [2].

Enclosed with this summary, are two articles that deals with the issue to create a soft keyboard that does not require a stylus or a dictionary and how to cope with the problem that a finger is occluding the target that is underneath it. This is done by suggesting a new technique for writing text on such devices. The overall research question for this study is: *How can we improve text entry on small touchscreen based mobile devices?*

The first article describes the technique (called directional selection) and evaluates it in a longitudinal user experiment. It deals with the question: *Can directional selection be used for text entry on small touchscreen based mobile devices?*

The second article takes on the perspective of an *implementation* of this technique (called QdQ) and compares it to several other existing text entry techniques. It deals with the question: *How does QdQ compare to existing small soft keyboards?*

2 Article Overview

2.1 Article 1

In this article we design and implement a new technique called Directional Selection, which is developed from a technique for selection of targets on maps [5]. It works by assigning a direction to each target, which then can be selected by pressing the finger down close to the target and dragging in the assigned direction. The idea is to transfer this idea to a technique that can be used for soft keyboards, since a keyboard on as small device basically also is a collection of small objects (the keys). We do the mapping by assigning one of four directions (left, right, up and down) to each key such that there is a maximum possible distance between two keys with the same direction. This layout can be seen in Figure 1(a).

(a) Arrow (b) Color

Figure 1: The QdQ Keyboards

This implementation of Directional Selection uses arrows on the keys to map a key to a direction. Another way of doing this mapping would be to use colors and then use the background to map the color to a direction, as shown in Figure 1(b).

To determine if the technique Directional Selection for keyboard design was a good idea, we used these implementations to perform a longitudinal user test. In this test, the 9 users were asked to write a series of sentences and homepage addresses over a period of 30 days. They did this on six different keyboards, where only the data for the two mentioned above is used in this article. During this test, the test application gathered information about exactly what the participants did on the keyboard (press down position, lift position, drag length, various timings etc.). This data were then

used to study how the keyboards performed, what we could improve and how they compared against each other.

The results from this test did show that Directional Selection is in fact a feasible technique. This is both in terms of the speed, error rate, mental overhead and the overhead time of the actual drag.

2.2 Article 2

In this article we changed the perspective from the *technique* to the actual *implementations* of the keyboard. In article 1 we determined that the Arrow version of the QdQ keyboard performed better in all scenarios, so naturally we used this keyboard in this experiment. The purpose of this article is to compare our keyboard to some of the existing techniques: QWERTY, QWERTY with dictionary, Multitap and Multitap with dictionary. This was to determine how it would perform against them in different conditions like sentences, homepage address, sitting and walking.

To do this comparison, we used the data from a longitudinal experiment, which was the same as described under article 1. In this article we use the data from five of the six keyboards (Arrow and the ones mentioned above). This data was analyzed to get empirical information about how fast the participants were with the different keyboards and how accurate they were. We also wanted to see what the differences were between homepage addresses that were not in the dictionary and normal sentences where all the words were in the dictionary. This is both from the dictionary perspective as well as from the distinct keyboard perspective.

The results from this test did show that the Arrow keyboard underperformed against the QWERTY keyboard in terms of raw speed. However, the error rates for QWERTY were much lower than those of similar studies, so there is need for more testing in this area to verify those results. Still, the error rate was lower on the Arrow keyboard, especially in the walking condition when entering homepage addresses. Against the other keyboards the Arrow keyboard did prove to be more versatile than the dictionary based keyboards (i.e. performed better in the walking and homepage address conditions) and had a higher speed than Multitap overall.

3 Empirical Study

This section documents how we did our empirical study in terms of implementing the keyboards and the research method we used.

3.1 Implementation

To do the experiment, we had to implement different 6 keyboards, a test program which presented sentences and homepage addresses for the participants and tools for analyzing the data. This section

provides an overview of the process of the implementation.

3.1.1 Keyboards

Implementing the keyboards was done in three iterations, this were not planned, but rather forced by the circumstances. In the beginning of the process we had two keyboards (Arrow and Color), which we were going to implement on the Android platform and compare to existing implementations of the other keyboards on other platforms. We started to implement our keyboards, but after some time the existing keyboards proved not sufficient in that they all had shortcomings (e.g. strange layouts for special buttons, not close enough to the standards, no real T9 implementation due to patents) and one big shortcoming in that to get all the keyboard types we wanted, we had to use three different platforms. Three platforms mean three implementations of the test application and a reboot of the phones every time we wanted to change keyboards. So it was decided to implement the keyboards ourselves, which would allow us to use the same test application and provide more control over the functionality of the keyboards.

So we extended the current implementation to include the other keyboards as well as an early version of a dictionary. However, having six working keyboards in a keyboard framework that was meant for only two, it was clear that we could not get the performance we wanted. The low performance was limiting the maximum possible writing speed too much. The only way to solve this was to restructure the whole implementation, basically resulting in a complete rewrite. The new structure was much more efficient and extend-able in terms of adding new keyboards.

Even though the keyboards now were efficient, the dictionary was not. It was meant as a prototype, so we had to rewrite this as well. However, fitting more than 25,000 words in the very limited memory on a smartphone as well as storing them so the (also limited) processor could check for about 1,500 words using a maximum of 50 milliseconds, was a huge challenge, which caused several rewrites of the data structures.

After finishing this system, it consists of 2,291 lines of Java code and 253 lines of XML code.

3.1.2 Test Application

Originally the experiment was divided into two separate experiments, with one for each article. Based on that we developed a test application which did not record any of the information that was needed for the second experiment. However, later on we merged the two experiments and therefore had to change several things in the test application to support recording of a much larger amount of data. Additionally, we had to support IPC (Inter-Process Communication) between the test application and the keyboards, for the data to be comparable and consistent. This was necessary because only the keyboard knows information about the actions on the keyboards (i.e. where the user presses, for how long, drag length etc.) and only the test application knows the progress of the user (i.e. which sentence, which user, which session etc.).

The development of the test application was done in several iterations, where the last one was in the middle of the evaluation. This change was code-wise large and involved a new way of storing the data, but was necessary, because the participants complained about the time the device had to use between two sentences to analyze and store data. It is important to note that this change did *not* interfere with the results, but only changed what happened *in-between* two sentences.

After finishing this system, it consists of 753 lines of Java code and 28 lines of XML code.

3.1.3 Tools for Analyzing the Results

After collecting the data, we had to analyze it. To provide an overview of how much data we are talking about, we can look at the numbers. On average each participant pressed a button on the keyboards about 34,000 times in total, resulting in 306,000 key presses for all nine participants. For each key press, we record 19 different numbers, which gives us a total of 5,814,000 numbers we need to analyze.

In the beginning of the evaluation, we were missing a way of keeping track of how well the participants were doing, but would not invest time in a system, that was useless afterwards. Therefore one of us used spare time to implement a homepage showing auto-generated graphs and tables with information of their progress. This tool proved very useful for the study, and later it was decided to use this as the tool for the final analyze of the numbers.

The graphs on this site were limited, because of how it was implemented in PHP, so we also had to generate CSV (Comma Separated Value) files, which we could use for generation of graphs in tools like OpenOffice Calc or Microsoft Excel.

After finishing this tool, it consists of 2001 lines of PHP code.

3.2 Research Method

Using the terminology described in [3], this project can be categorized as using Laboratory experiments as research method with engineering as research purpose. Laboratory experiments are characterized by taking place in a controlled environment, which does not necessarily have to be in a laboratory, but could be in an office or hallway. Compared to field studies, which is conducted in the real world, it has the following advantages:

- Control of variables: In a controlled environment it is easy to control what variables the participants are exposed to before and during the test.
- Replicable: Given that you have more control over the variables, it is also much easier to replicate.

- Easy data collection: The test environment is usually under the control of the researchers, which allows for easier integration with data collection equipment like cameras, microphones, computer programs etc.

And has the following disadvantages:

- Limited realism: Depending on what is being studied, the laboratory usually lacks the interruptions and other variables of the real world, which might impact the results.
- Unknown generalizability: It is hard to tell if a given results will be the same in the real world, because the laboratory setting does usually not account for the variables that might influence the results elsewhere.

Laboratory experiments were chosen because we want some control over how and when the participants wrote their sentences / homepage addresses. The test was not carried out within a laboratory and not even at a place we could directly control. Instead we told the participants how to sit and that they should do the writing in a quiet place where they could concentrate. This approach was chosen because the large amount of typing hours would require a lot of planning if we had to watch over them. Additionally, we prioritized that they should be able to do it when they had time and in parallel, which would not be feasible using laboratories.

The choice of this research method, does most likely impact the results in that we probably would get a lower speed and higher error rates in the real world, with all the interruptions. However, doing it in the field, the results would be inconsistent, because the performance of the participants would depend on the current environment they were in. So with laboratory experiments it is much more clear what works and what does not.

4 Conclusion and Future Work

In this master thesis we designed a new text entry technique based on an existing technique for basic target selection on maps. This technique, called Directional Selection, was examined from two different perspectives. The first perspective was from the technique itself, where we implemented two variations of this and tried to determine if it is feasible. The second perspective was from the keyboard implementation of this technique, where we compared it to several other existing keyboards. The data used in both perspectives (articles), originated from the same longitudinal user test where we collected a vast amount of data from 9 participants writing a series of sentences and homepage addresses over a period of 30 days.

In article 1 we determined that the technique itself has a lot of potential, with a low overhead in time, low error rate, good performance and a workload similar to the usual keyboards of the participants.

From article 2 we know that the implementation of Directional Selection (Arrow) had a lower speed than a standard QWERTY, but also lower error rate. However, this result is very different than that of similar studies, because the QWERTY error rates should be a lot higher. The Arrow keyboard did prove to be more versatile than the dictionary based keyboards and overall performed much better than Multitap.

In the introduction, we asked the question: *How can we improve text entry on small touchscreen based mobile devices?*. This has been answered through the design, implementation and evaluation of Directional Selection, because it has the potential to improve soft keyboards. The current implementation is troublesome because the participants were not fast enough to do the mapping from arrow or color to direction (50% of the overhead is this mapping). This could be improved in future implementations, by using another layout where it is more obvious. This could be Multitap, where we exploit the large buttons, by placing the letters such that their position on the button, relative to the middle, symbolizes the dragging direction to get that letter. The point is that the technique has proved feasible, but to gain the performance necessary to replace existing techniques, it should be attempted implemented on other layouts that QWERTY.

4.1 Limitations

The following limitations should be considere when using the results from this paper:

- The surface of the screen were a bit resistant when it comes to dragging compared to the default screen, which might cause a lower performance when dragging.
- The screen does only support single-touch, where a multi-touch is becoming more and more popular. On a multi-touch screen the numbers could be very different.
- The participants were not instructed in how long a break they should have between sessions. Too long breaks did lower the performance because they got out of the routine, and too short did also lower the performance, because they got tired. This is a result from the experiment.
- We did only have one walking session, and based on the difference between the sessions before that one, the results may not be accurate.
- The results may not be the same ”in the real world”, since they are based on laboratory experiments.

References

[1] Tegic. `http://www.nuance.com`.

[2] COCKBURN, A., AND SIRESENA, A. Evaluating mobile text entry with the fastap keypad. In *HCI 2003* (2003), British HCI Group, pp. 77–80.

[3] KJELDSKOV, J., AND GRAHAM, C. *A Review of Mobile HCI Research Methods*. In Proceedings of Mobile HCI 2003, 2003.

[4] MACKENZIE, I. S., KOBER, H., SMITH, D., JONES, T., AND SKEPNER, E. Letterwise: prefix-based disambiguation for mobile text input. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2001), ACM, pp. 111–120.

[5] YATANI, K., PARTRIDGE, K., BERN, M., AND NEWMAN, M. W. Escape: a target selection technique using visually-cued gestures. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2008), ACM, pp. 285–294.

[6] ZHAI, S., AND KRISTENSSON, P.-O. Shorthand writing on stylus keyboard. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM, pp. 97–104.

QdQ: Exploring Directional Selection Techniques for Small Soft Keyboards

Anders Houbak Kristiansen
Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
DK-9220 Aalborg East, Denmark
andershkristiansen@gmail.com

Frederik Larsen
Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
DK-9220 Aalborg East, Denmark
flabby@cs.aau.dk

ABSTRACT
This study investigates the use of directional selection as a technique for implementing a new type of soft keyboard for small mobile devices. This technique is used to implement two different keyboards, with one based on arrows and one on colors. These are then compared against each other using the results from a longitudinal user test to determine if directional selection is feasible as well as which of the two designs that are best. Our results show that directional selection is a feasible technique for use with soft keyboards.

Author Keywords
soft keyboards, text entry, touchscreen, QdQ, directional selection

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Input devices and strategies*

INTRODUCTION
Small mobile devices are used in many new areas where the desktop or laptop computer used to dominate, like browsing homepages, writing emails and participating in social networks. All of these new requirements require them to support the users in typing text with a decent efficiency in many different contexts. This is what this paper will try to improve, by introducing a new technique and evaluating it in a longitudinal user experiment.

One of the goals of this technique is to be independent on dictionaries, which has two important drawbacks we want to avoid. The first is, as [11] shows, that if only 15% of the words written is not in the dictionary all speed increase are lost. This can be words like abbreviations, usernames, passwords, email and web page address, which all are very important for the new areas the smartphones are used in. The other reason is the high cognitive load that the users feel when they are writing, because they need to focus on what the dictionary shows on the screen and on the next key they need to press [8].

Another goal is to avoid using a stylus. Many new smartphones, like the iPhone from Apple and the HTC Touch Diamond 2, uses a capacitive touchscreen that works by utilizing the small amount of electricity the human body conduct. This means they will not work with a standard stylus. Another reason for not using a stylus, is that the intense attention required can disrupt the users attention from the actual text input and it can feel tedious using a stylus over an extended period of time[21].

When we do not have a stylus or dictionary to help us hit the desired key, we have to find another way of doing so. This problem is illustrated in Figure 1, where it is near impossible to identify the key pressed from the position of the finger.

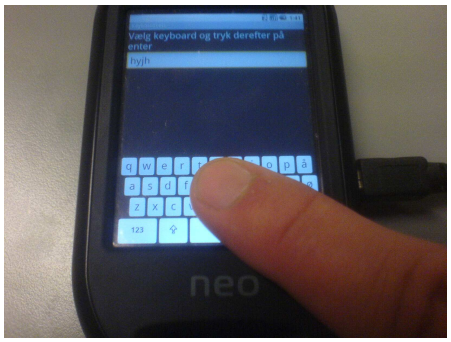


Figure 1. The problem with a finger on a small soft keyboard.

To solve this problem, we have developed a new technique called Directional Selection, which will be evaluated through this paper.

This paper deals with the following question: *Can directional selection be used for text entry on small touchscreen based mobile devices?*

RELATED WORK
When implementing a keyboard, the size of the buttons is a very important thing to consider. Previous research shows that soft buttons needs to have a size of 22mm to be useable[10][13], but newer studies has shown that even smaller

Submitted for review to CHI 2010.

buttons still can be useable. [16] and [9] shows that when buttons becomes smaller than 10x10mm, the entry speed decreases and error rate increases. One example of this is as [9] shows, going from buttons with the size of 13.8x9.8mm to 7.2x6.5mm the CPS(Characters per Second) went from 2.65 to 1.80 and 4.4% errors to 11.1%. Another example is [16] that comes out with an error rate of approximately 46% with a button size of 5.2x5.2mm. This was by using the tip of the finger, but when they used the nail, the error rate dropped to 30%. Increasing the button size to 10.5x10.5mm resulted in the error rate dropping to approximately 2-4%. But what can you do, when you do not have the screen space available to place 26 buttons larger then 10x10mm on the device?

VirHKey is a gesture based keyboard made to be used with stylus, where each key consist of 1-4 small strokes (dragging a small distance) where each stroke can be in one out of 5 different direction. In average each letter is 2.04 stroke, VirHKey work by there is a circle that shows the letters you get when dragging in a direction as seen in Figure2 and each time a stroke is made the circle change to show what you get the next time. They got 25.8WPM for expert users and from a SUS-like questionnaire showed the participants had a good satisfaction with using this keyboard.

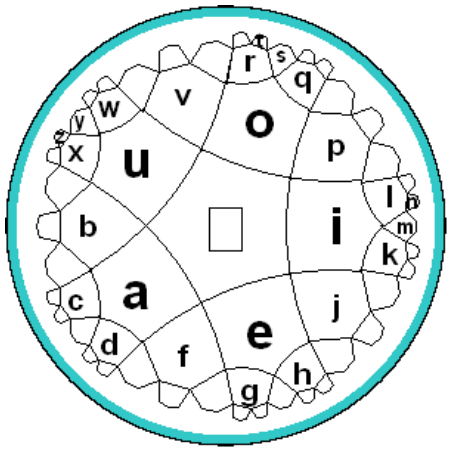


Figure 2. The VirHKey keyboard.

[18] have created a mobile phone that uses a chorded Multitap keyboard and a combination of three buttons on the back (chords) together with the normal keys to select which letter to write. If the first cord is pressed, the first letter is selected on the Multitap key and the second chord selects the second letter and so on. Using the chorded Multitap keyboard was significantly faster than Multitap in their experiment. [20] also created a keyboard that is using chords, where each chord shows a distinct third of a QWERTY keyboard.

Instead of using chords to select which letter to write, an accelerometer can be used like in [17], where the mobile phone is using an accelerometer to see when it is tilted. If it is tilted a little to the left, the first letter on the button is selected and if it is tilted a lot the upper case version is selected. Tilt up is the second, left the tried and down the fourth. By using this technique in their experiment, they got a significantly faster entry rate than with Multitap.

A technique that focuses on a combination of high speed and accuracy when selecting small targets on a touchscreen, rather than writing speed, is Escape[19], that is used to select small targets which was occluded by the finger that was selecting. Escape works by giving all the objects a direction and to select an object the user simply drags the finger in the direction of the object. This is illustrated in Figure 3. One of the problems with Escape is that it does not work when many targets are close together because it then cannot map all of them to a few different directions and still be possible to select them. There is also a limitation on object density at the edge of the screen. For instance, it is not possible to drag down if the object is at bottom of the screen, but if you drag down just above it, it will still be selected, if it is the closest object with that gesture.

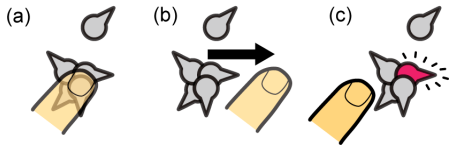


Figure 3. The Escape selection technique.

OUR TECHNIQUE: DIRECTIONAL SELECTION

The idea is to modify Escape that was described in Related Work, such that it can be applied to a soft keyboard. Doing this could help us achieve:

1. Independence of dictionaries, which would make it easier to write words that are not in the dictionary (like abbreviations, URL's, second language words and email addresses). As shown by [11], the dictionary actually *lowers* the writing performance if only 15% of the words are not in the dictionary. This is feasible due to a lower error rate.
2. Emulation of a larger keyboard, since the area in which you need to hit to select the desired key, is much larger than the keys, making it more feasible to use the fingers instead of a stylus.
3. Lower stress level, since the keyboards that are dependent on a dictionary or a stylus are known to have a high mental demand [8][21].

In related work, we mentioned two drawbacks regarding Escape when used on maps. By transferring the technique to a soft keyboard, we can avoid these by choosing an effective number of directions and mapping of these on the keys. The resulting technique has been named Directional Selection and a keyboard implementation of this, named QdQ, can be seen in Figure 4. It uses four directions, which is the minimum number necessary to avoid having two keys with the same direction right besides each other. Additionally, the mapping was carefully chosen such that you never have to drag against an edge to select a key.

The use of arrows to map a key to a direction is the obvious way to implement this technique, but it is not the only

Figure 4. The QdQ keyboard in Arrow Mode.

one. Figure 5 shows another implementation of the idea using colors instead of arrows, where the background color is used to map a button color to a direction. This could be a good idea, because humans are a bit faster to recognize colors compared to shapes and has a lower error rate using colors[14]. Both of these implementations will be evaluated through this paper, where they will be refered to as Arrow (Figure 4) and Color (Figure 5).

Figure 5. The QdQ keyboard in Color Mode.

Note that the layout is in Danish, which is because the test participants are from Denmark and should write in their primary language to avoid spelling trouble and other language related slowdowns. The SET button is used to switch between this and the other keyboards we have implemented for the evaluation, which are described in more details in the following sections. It is important to understand that the idea is not bound to QWERTY layouts, but can be applied to any layout (like Dvorak [7], Fitaly [3] or OPTI [12]).

As an example of usage, to write "hey", you need to follow this procedure:

1. Press down on letter `h`, move your finger a bit right and lift your finger again
2. Press down on letter `e`, move your finger a bit up and lift your finger again
3. Press down on letter `y`, move your finger a bit left and lift your finger again

It uses the position you press down to determine the center of the area to look for the letter and the position where you lift your finger to determine the direction. The area around the down-press is much larger than the actual keys, since the keyboard always chooses the closest key with the detected direction. Consequently, it actually emulates a larger keyboard, because you can press anywhere in that area and still get the desired key.

Regarding the performance of the technique, we can present the following hypotheses:

1. The performance on both keyboards improves over time as the user learn the technique.
2. Arrow will be faster than Color in the beginning, because the mapping from color to direction takes more time to learn, but Color will be faster in later sessions.
3. The error rate of Color and Arrow will be about the same, since the action required to write a letter is the same.
4. The performance of the individual key depends on the number of times it has been used, because the user is more likely to remember the mapping.
5. The time it takes to write one character is higher for both Color and Arrow than on a keyboard without dragging, since the user needs to perform one more action (the drag).

METHOD

To evaluate our technique, a comparative longitudinal user test using within-subjects was performed using the method described in this section. Each user did 19 sessions with two keyboards where they wrote both sentences and homepage addresses.

Participants

12 participants participated in this test but only 9 finished. Six out of the 9 participants were from Aalborg University, the last three participants were from outside the university. All 9 participants were volunteers and did not get paid for participating. Two of the participants where females and seven males, they had an age range from 21 to 49 (mean 28) years old. One of the participants was left-handed and none of the 9 participants were color-blind.

Six of the 9 participants use a mobile phone with a T9 keyboard with physical keys almost every day, where only two uses Multitap with physical keys and none of these participants had any experience when it comes to writing on a touchscreen based device. The last participant had some experience using a touchscreen based mini QWERTY keyboard on a smartphone. All of the participants were experienced when it comes to writing on a standard personal computer QWERTY keyboard.

Apparatus

The test was performed on two Openmoko Freerunner smartphones running Android with 128MB SDRAM and a 400MHz ARMv4T processor. It has a resistive touch screen with the dimension 480x640 pixels (1 pixel = 0.09mm), that does not support multi-touch, which implies writing with only one finger. The touchscreen was covered with a protective film that added more resistance when dragging than the standard touchscreen. The test program and keyboard was programmed in Java using the Android API over a period of 3 months. There was a high focus on little resource usage and a good performance, to ensure that it was not a bottleneck. A screenshot can be seen in Figure 6 with the arrow based QdQ keyboard.

Figure 6. The Arrow keyboard.

The test was performed with two different conditions, QdQ and QdQ with colors.

The QdQ keyboard with arrows, is the implementation of the idea presented in [19], where the button layout is a normal Danish alphabet QWERTY keyboard using arrows to give the participant a mapping from button to direction. It can be seen in Figure 6. The 123 button gives access to the numbers and symbols and the set button is used to change between the five conditions(will be disabled under the test, to prevent participants to accidently pressing it). The size of the buttons containing letters were 3.8x4.6mm, where the enter and shift buttons had a size of 6.5x5.8mm, ”123” and delete buttons were a little larger with a size of 8.6x5.8mm and the space button with a size of 13.0x5.8mm. A minimum drag size of 1.35mm was chosen from a pilot study as the size, where a drag was long enough to not just be a simple click. The only way to enter letters was our by technique and not by clicking, the other buttons (bottom row) was still working like in a normal QWERTY keyboard as they needed to be clicked on. This was done to prevent the inconvenience it would be to drag down to the bottom of the screen since there is an edge and the pilot study showed that there was no problem doing it this way.

The QdQ keyboard with colors is the same as QdQ with arrows, but instead of using arrows as a mapping from button to direction, it uses colors.

We will use Arrow as short name for the QdQ keyboard with arrows and Color for the QdQ keyboard with colors.

Experimental Design

Within-subjects were used for the test with 19 sessions as factor crossed with the two conditions QdQ and QdQ with colors. There were six conditions in the actual test, but only two of those are used in this article. The dependent variables are entry rate (Words per Minutes and time used in different touchscreen usage) and accuracy (coordinates and error rates of touchscreen usage).

To prevent carryover effects, the keyboards were distributed in a 6x6 balanced roman square where participant one started with the first row, then second row next session. Participant two started with the second row and so on for all the participants. This was done to mix the order of the keyboards. All of the sentences were picked randomly (but distinct for the specific sessions) from a set of 602 sentences. They only contained lower case characters. All of the sentences were obtained from three different short stories[5][6][4]. The sentences had a length between 16-44 characters (with an average of 28.5), consisted of 3192 words (1084 unique words) and the average word length were 4.6 characters. The homepage addresses were randomly selected from the 100 most popular homepages in the world[1] without the ”http://www.”. They contained lower case characters, numbers, ”-” and ”.”. The order of the eight sentences and the three homepage addresses changed from keyboard to keyboard and also from session to session by alternating between them. The words in the dictionary was from [2] where all the words from the sentences were added this was done to enable us to look at the data from the sentence as 100% represented in the dictionary. None of the homepages addresses was in the dictionary to give us opposite data, 0% representation. It was decided that every participant started with their own empty learning which they kept building on throughout all of the sessions. This was decided to be the most fair, and resemble the real world the most, compared to no learning or perfect learning.

Procedure

Before the first session the participants got the chance to write a few sentences with all of the different keyboards and afterwards ask questions if there was something they did not understand. They were instructed to write as quickly and accurately as possible, where they were allowed to have uncorrected errors. Sentences were instructed to be read meticulously before stating to write, to keep FOA (focus of attention) down. The sentences remained visible under the test to prevent spelling errors and prevent them from forgetting the sentences. In each of the sessions they had to write eight sentences and three homepage addresses with each of the keyboards. After they were finished with one keyboard they had to change the keyboard on their own and continue the session until they were done, with all of the keyboards. There were no restrictions on where they were located under the test, but they were asked to make sure they sat comfortably and undisturbed before starting. Every time a participant was finished with a couple of sessions using the smartphone they had to bring it to us so we could take a backup and transfer test data from another participant to it. This was a necessary such the two smartphones should be used by the seven participant in the same time frame. They also had the opportunity to borrow one of the smartphones home at the evening or at the weekend to test. Seven of participants had a deadline of 20 days after the test began to complete the sessions however this timeframe had to be extended to 30

days because only one was finished at the deadline. The last three participants got a seven days deadline to complete all the sessions. How the sessions were distributed was up to the individual participant. Pauses could be held between two sentences, between the different keyboards and at the end of each session. After all of the sessions we asked the Danish version of the following questions to the participants, while recording their responses:

1. What do you think is good about the Arrow Mode keyboard?
2. What do you think is bad about the Arrow Mode keyboard?
3. What do you generally think of the Arrow Mode keyboard?
4. What do you think is good about the Color Mode keyboard?
5. What do you think is bad about the Color Mode keyboard?
6. What do you generally think of the Color Mode keyboard?
7. Is there anything you think we could improve?
8. How does the Arrow Mode keyboard compare to what you are used to for writing on a mobile device?
9. How does the Color Mode keyboard compare to what you are used to for writing on a mobile device?
10. Any comments?
11. Any problems during the test?

Data Collection

For each key pressed, we collected a time stamp which started from when the first key was pressed in a sentence, key code (including space and backspace), session number, sentence number, keyboard used and a reference to the written sentence which was used to help find all the uncorrected errors. At the same time all the coordinates where the participant pressed down and lifted the finger were recorded together with the elapsed time. All test data was automatically collected on the mobile phone and saved to a mini SD Card where all of the participants data were stored separately.

Data Analysis

The data that was collected was used to find learning effects, WPM and error rates. WPM is calculated as the time used to type the sentence plus the average time to correct an error multiplied with the number of uncorrected errors in the sentence. The combined time is divided with the number of characters in the sentence including spaces minuses the first character. This time was then divided by 60 to get it in minutes. Finally this is divided with five as the accepted word length to get WPM.

To measure the error rates, we use corrected, uncorrected and total error rate as described in [15]. We define CPS as Character per Second, without any error calculations included. For statistical analysis, we use the F-test throughout this paper.

RESULTS

In this section, we will show the results from the test which relates to answering the research question. Additionally, we will prove or disprove the hypotheses presented previously in this paper.

Entry Rates

In Figure 7 the Color and Arrow keyboard are compared in terms of WPM when entering sentences. The Arrow keyboard has the highest WPM in the beginning with 9.4 compared to 7.5 for Color. The average difference between the two keyboards is significant ($F_{1,16} = 30.8, p = 0.0005$). Both Arrow and Color benefits from learning effects and have a WPM increase from session 1 to 19 of 60.2% and 69.9% respectively.

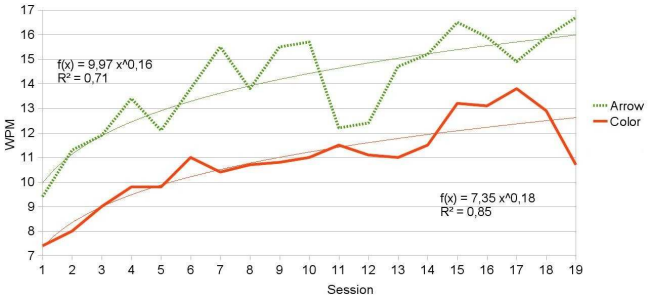


Figure 7. Comparing Arrow and Color WPM (sentences)

The WPM when entering homepage addresses is shown in Figure 10. It shows that Color actually has a higher or equal WPM in two sessions (8 and 18), but Arrow still has the highest average WPM (13, compared to 10.7 for Color). The difference between the two is significant ($F_{1,16} = 23.72, p = 0.0012$). Additionally, both show a clear learning effect.

Looking at the WPM when typing sentences and homepage addresses combined, enables us to answer hypothesis 1 and 2:

Hypothesis 1: The performance on both keyboards improves over time as the user learn the technique.

Answer: The learning curves for sentences shows an improvement of 60.2% and 69.9% for Arrow and Color respectively, which tells us hypothesis 5 is true.

Hypothesis 2: Arrow will be faster than Color in the beginning, because the mapping from color to direction takes more time to learn, but Color will be faster in later sessions

Answer: Color is significantly slower than Arrow in all sessions, but the learning curve shows that Color most likely will stay under Arrow, so hypothesis 1 is false.

Accuracy

The error rates for the two keyboards while writing sentences can be seen in Figure 9 and the error rates for homepage addresses can be seen in 10. The average total error rate over all sessions for Arrow is 5.5% (3.7% corrected and

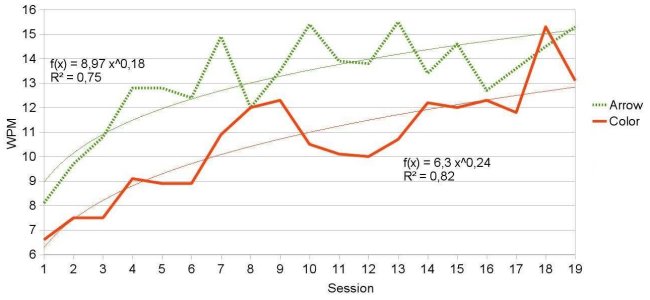


Figure 8. Comparing Arrow and Color WPM (homepage addresses)

1.8% uncorrected) and 6.8% for Color (5.1% corrected and 1.7% uncorrected).

The difference is significant for sentences ($F_{1,16} = 11.22, p = 0.0101$), but not for homepage addresses, which enables us to answer hypothesis 3:

Hypothesis 3: The error rate of Color and Arrow will be about the same, since the action required to write a letter is the same.

Answer: It is false, because there is a significant difference between Color and Arrow when writing sentences. However, the hypothesis does hold for homepage addresses.

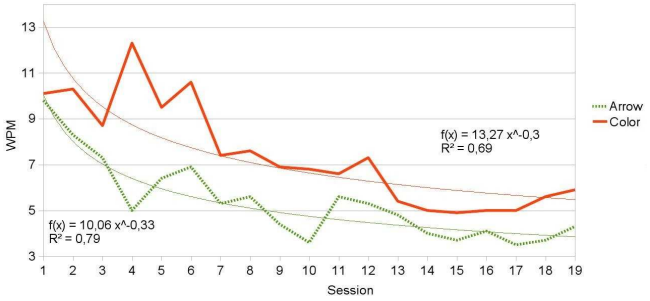


Figure 9. Comparing Arrow and Color error rates (sentences)

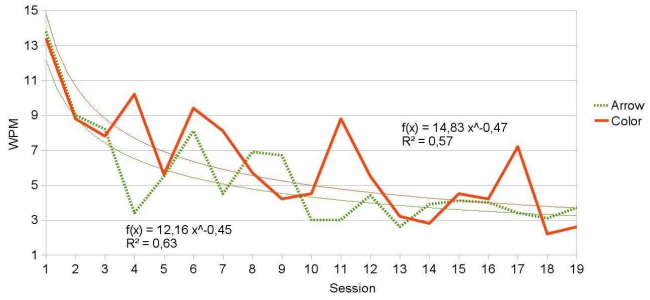


Figure 10. Comparing Arrow and Color error rates (homepage addresses)

Both sentence and homepage address writing benefits from learning effects. The Error rate of the first session is 9.8% for Arrow and 10.1% for Color. For the last session it is 4.3% for Arrow and 5.9% for Color, which is 56% and 42% lower than the first session respectively.

Character Distribution

Figure 11 shows the CPS for the Arrow keyboard, ranging from 0.9 CPS (c) to 1.5 CPS(e). The data does not show any consistent sign of there being a difference between areas (i.e. sides, center, bottom etc.) of the keyboard in terms of entry speed. The fact that e is the character most represented in our sentences and also has the highest CPS (1.5), could suggest that the speed depends on the frequency of the given letter. This is even easier to see, when comparing Figure 11 with Figure 12, which shows the frequency on the actual keyboard.

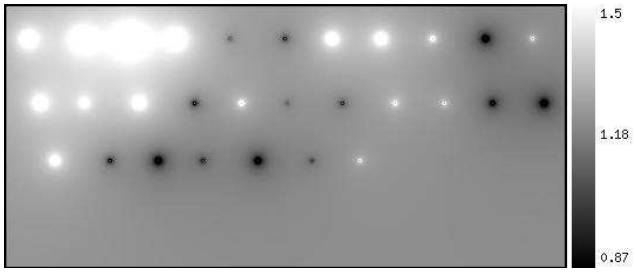


Figure 11. The CPS distribution over the Arrow keyboard (white is high and black is low). To map to the keyboard simply imagine placing this figure on top of Figure 4 (the Arrow keyboard).

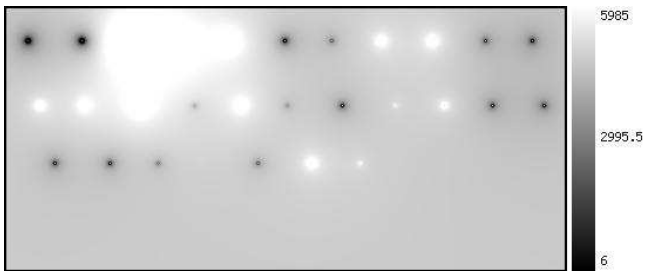


Figure 12. The letter frequency distribution over the Arrow keyboard (white is high and black is low). To map to the keyboard simply imagine placing this figure on top of Figure 4 (the Arrow keyboard).

Figure 13 and Figure 14 shows a graph over CPS and character distribution respectively. The characters in the figure are ordered by frequency, with the left being the least frequent and the right being the most frequent. Notice that the order of the letters are different in the two figures, which is because the frequency is based on how frequent the given letter is with *the specific keyboard* and *not* overall.

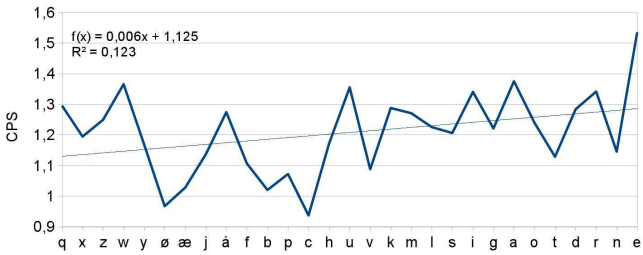


Figure 13. CPS by key (Arrow)

Hypothesis 4: The performance of the individual key depends on the number of times it has been used, because

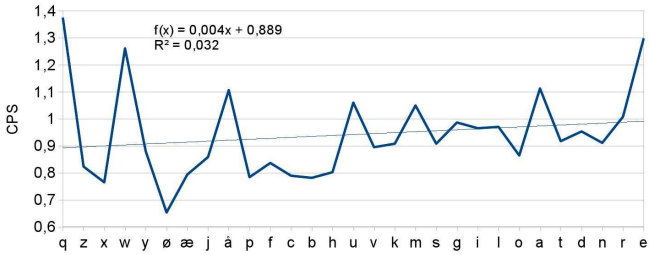


Figure 14. CPS by key (Color)

the user is more likely to remember the mapping.

Answer: The results were not consistent enough for it to be significant, but the graphs does suggest that it might be true.

Dragging

Overhead

Based on numbers from the other four conditions in the test, we know that the average time to write a letter on a normal button is 990ms. The average time for typing a letter on the Arrow keyboard, were 1106ms. The average time that the participants held their finger on the screen while dragging, were 203 ms for Arrow. On a normal button it is 144ms. This tells us that the overhead for dragging is 59ms for each character written. When compared to the 990ms a click takes with a normal button, the dragging consequently adds 5.9% overhead in time.

The total overhead of Directional Selection is 116ms (1106-990). If we subtract 59ms for the actual drag, we get a 57ms remainder, which can be considered the ”mental overhead”. This means that half of the overhead, between our technique and just clicking on a normal button, is to identify which direction to drag.

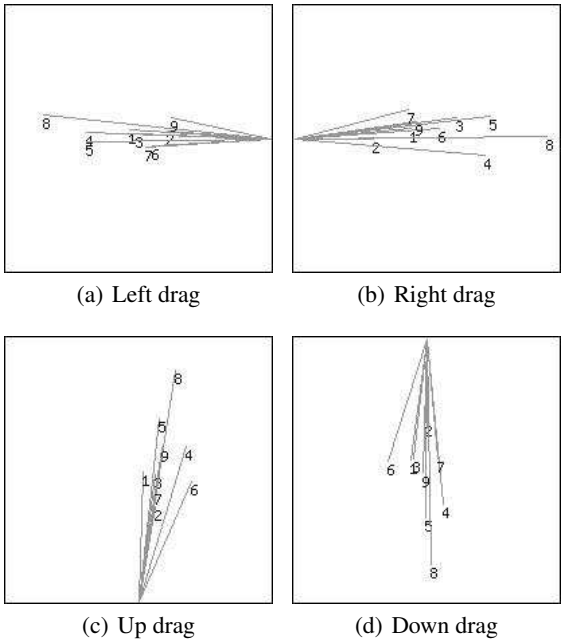
Hypothesis 5: The time it takes to write one character is higher for both Color and Arrow than on a keyboard without dragging, since the user needs to perform one more action (the drag).

Answer: Being 116ms slower, where 59ms is for the actual drag and 57ms is the mental overhead, means that this hypothesis is true.

Directions

Figure 15 shows the average drag directions and length, for each for the four directions. The numbers at the lines maps to a participant ID from 1 to 9. The important result here is that all participants are far from the critical 45 degree (for which you will get a wrong direction). There is no clear evidence of a difference between our left handed participant (9) and the others.

The participant with the shortest average dragging length were number 2 (2.03mm) and the longest were number 8 (5.08mm). So with only 1.35mm necessary for a successful drag, participant 8 is dragging 276% longer than necessary.



(a) Left drag (b) Right drag

(c) Up drag (d) Down drag

Figure 15. Dragging directions by participant. The width of the figures correspond to 6mm or 66.7 pixels)

Average dragging time for participant 2 and 8 were 117.7 and 209.8 respectively. This means that the dragging time of participant 2 was 78.2% faster than participant 8. This suggests that if the participants learned to shorten their drag, they would be faster.

Data Summery

If we look at the results so far, it is clear that Arrow is outperforming Color in entry rate and they are about equal in their error rates. The dragging concept is also working without any problems or large overhead. To compare a keyboard using Directional Selection, to one without, we can try to look at the average cost of deletion (i.e. the time it takes to press delete and enter a new character) which is 2,474.5ms on Arrow. So, with an average total overhead of 116ms per character, for the dragging to be worth it, it has to ”fix” one error for every 21.3 characters entered. In other words, the error rate has to be 4.7% lower on Arrow than on normal buttons. The total error rate for Arrow were 4.3% in the last session, which is more than 4.7% less than the 11-30% similar studies have shown for normal buttons of that size[16][9]. This tells us that Directional Selection can be a used in applications where there is a high error rate and accuracy is important.

Questions

Generally the answers related to Arrow or Color was answered with the same answer, except that almost everybody said that the mapping from color to direction was much harder than the arrow to direction. This resulted in they were sure that they performed worse on the Color keyboard. The evaluations of the two keyboards (questions 1-6) had the following in common:

Good: All participants felt that it was easy to hit the desired key. Additionally, some said that after more practice they could easily see themselves use Arrow as their primary mobile keyboard.

Bad: The first item in the view field at the left side of the screen was hard to hit, because of the high edge around the screen.

When asked if we could improve anything, some of the participants said that the time limit (combined with their own deadlines) put a pressure on them to get finished, and if they could schedule the test period themselves, then they could perform better and more consistent. Also we got a comment from one of the participants that she felt that the predefined text was putting additional mental pressure on her. She argued that if she could get to use our device as her everyday phone and write what she wants, then she would have more energy for the actual typing.

When asked how the Arrow keyboard compares to the one they are used to, they generally was positive about the Arrow keyboard, but most of them would like to continue with their usual keyboard.

When asked how the Color keyboard compares to the one they are used to, they *all* said that they liked Arrow much better, and never would use Color themselves.

None of the participants had any comments or any problems during the test.

DISCUSSION

The accuracy and performance differences between Arrow and Color, are a result of the different mapping between keys and direction in the two keyboards. The Arrow keys have a direct mapping between key and direction, where the Color keys have an indirect mapping between key and direction. This should result in Arrow being faster than Color for new users. Then the question is whether people can learn the mapping between colors and direction, so it comes naturally to them. If that is possible, then Color should be the fastest, because humans are faster to recognize colored objects then shapes[14]. The data shows that the mapping from color to direction is quite time consuming, even after 19 sessions. This is also supported by the answers from the participants when they were asked questions about the Color keyboard. One said:

> It was really difficult to remember the direction for each color, so you had to take a break while you were writing to get an overview of which direction each color has. This is very time consuming

Similar comments came from eight of the participants. This tells us that using colors is not a great idea even if they are recognized faster because the mapping takes longer than the time gained. This supports the observation made in Escape[16], where they did not see increased speed in selection by adding color to the arrow shaped objects.

As shown before, there does not seem to be difference between the areas of the keyboard in terms of entry speed. However, the downtime distribution in Figure 16 shows that the participants held their fingers down longer in the sides of the device, compared to the center. Actually, the difference between a button in the left side (a) and a button in the middle (g), which both have the same direction assigned, is significant ($F_{1,3794} = 14.43, p = 0.0002$). This does not necessarily map directly to a lower entry speed, since the additional time used on the current key could be used to prepare for the next. However, it still could indicate that some of the participants had trouble dragging in these areas.

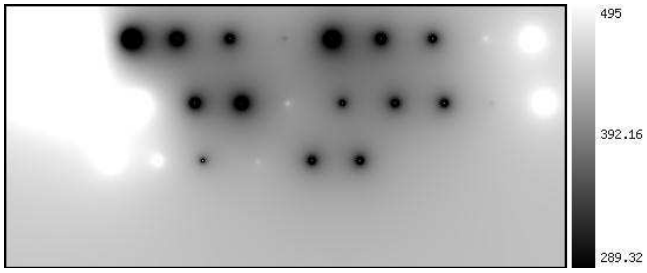


Figure 16. The downtime distribution over the Arrow keyboard (white is long and black is short). To map to the keyboard simply imagine placing this figure on top of Figure 4 (the Arrow keyboard).

When looking at the difference between left and right handed participants in downtime(the time between the user press down and till he/she lifts the finger again), we saw a clear difference. The time the right handed participants used on downtime, can be seen in Figure17. It shows that they used more time on the all the letters in the right side of the keyboard than those in the middle and left. If this is compared to the one participant that is left handed, shown in Figure18, it is almost mirrored. This shows that participants have a longer downtime in the same side of the keyboard as the hand they use to write. This is most likely caused by the high edge around the screen that forces the participant to turn the hand such the finger is underneath it and thus losing some of the control of the finger. But the time lost with the longer downtime is so small that it has no visible effect on the CPS, which tells us that the edge on the phone has no real impact on writing speed.

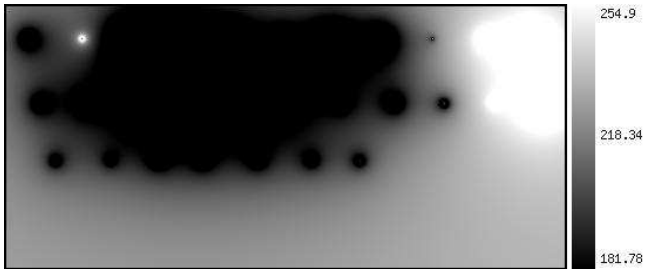


Figure 17. The downtime distribution over the Arrow keyboard for right handed (white is long and black is short). To map to the keyboard simply imagine placing this figure on top of Figure 4 (the Arrow keyboard).

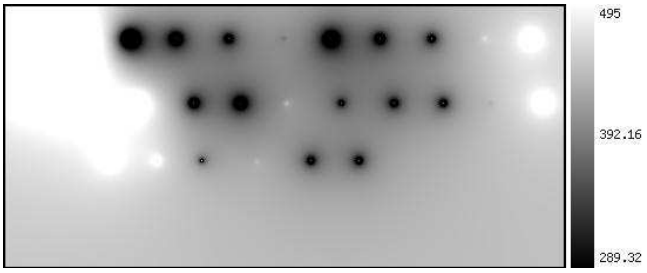


Figure 18. The downtime distribution over the Arrow keyboard for left handed (white is long and black is short). To map to the keyboard simply imagine placing this figure on top of Figure 4 (the Arrow keyboard).

CONCLUSION

In this article we attempted to answer the following question: *Can directional selection be used for text entry on small touchscreen based mobile devices?*

To answer this, we implemented two variations of this idea. The first one, called Arrow, used arrows on the buttons of the keyboard to map a button to a direction. The other, called Color, used colors on the buttons, which mapped to one of four colors to a identical color on the background. The color on the background was placed such that the user needs to drag in that direction to choose the intended key.

These two keyboards were then tested in a longitudinal user test, where 9 participants were asked to write 8 sentences and 3 homepage addresses with each keyboard 19 times.

Based on the results we can answer the research question with a definite "Yes". The overhead of the drag were tested to be 116 ms, or 4.7%, which is low compared to the error rate reduction potential. The error rate of Arrow were under half of what studies have shown the error rate to be on keyboards without Directional Selection.

Limitations

The following limitations should be considere when using the results from this paper:

- The surface of the screen were a bit resistant when it comes to dragging compared to the default screen, which might cause a lower performance when dragging.
- The screen does only support single-touch, where a multi-touch is becoming more and more popular. On a multi-touch screen the numbers could be very different.

Future Work

Half of the overhead, between our technique and just clicking on a normal button, was to identify which direction to drag. This suggests that any improvements in the mapping could make a big impact in the overhead for dragging. So Directional Dragging should be attempted implemented on other layouts than QWERTY, which might result in a better performance, if the mapping could be made more obvious.

REFERENCES

1. Alexa the web information company. `http://www.alexa.com/`.
2. Danish wordlists. `ftp://ftp.mathematik.uni-marburg.de/pub/mirror/openwall/wordlists/languages/Danish/`.
3. The fitaly one-finger keyboard. `http://www.fitaly.com/fitaly/fitaly.htm`.
4. Rejse mod vinter. `http://www.fyldepennen.dk/tekster/29289`.
5. Sort og hvidt hjerte. `http://www.fyldepennen.dk/tekster/37687`.
6. Uden titel. `http://www.fyldepennen.dk/tekster/38526`.
7. BROOKS, M. Introducing the dvorak keyboard. `http://dvorak.mwbrooks.com/`.
8. COCKBURN, A., AND SIRESENA, A. Evaluating mobile text entry with the fastap keypad. In *HCI 2003* (2003), British HCI Group, pp. 77–80.
9. LEE, S., AND ZHAI, S. The performance of touch screen soft buttons. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems* (New York, NY, USA, 2009), ACM, pp. 309–318.
10. LEWIS, J. R. Literature review of touch-screen research from 1980 to 1992. In *IBM: Design Center/Human Factors* (Boca Raton, FL, USA, 1993), IBM, pp. 1–6.
11. MACKENZIE, I. S., KOBER, H., SMITH, D., JONES, T., AND SKEPNER, E. Letterwise: prefix-based disambiguation for mobile text input. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2001), ACM, pp. 111–120.
12. MACKENZIE, I. S., AND ZHANG, S. X. The design and evaluation of a high-performance soft keyboard. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1999), ACM, pp. 25–31.
13. MARTIN HELANDER, THOMAS K. LANDAUER, P. V. P. Handbook of human-computer interaction. Elsevier Science Pub Co.
14. QUINLAN, P. T., AND HUMPH, G. W. Visual search for targets defined by combinations of color, shape, and size: An examination of the task constraints on feature and conjunction searches. In *Perceptton & Psychophysics* (1987), pp. 455–472.

15. SOUKOREFF, R. W., AND MACKENZIE, I. S. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM, pp. 113–120.

16. VOGEL, D., AND BAUDISCH, P. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2007), ACM, pp. 657–666.

17. WIGDOR, D., AND BALAKRISHNAN, R. Tilttext: using tilt for text input to mobile phones. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2003), ACM, pp. 81–90.

18. WIGDOR, D., AND BALAKRISHNAN, R. A comparison of consecutive and concurrent input text entry techniques for mobile phones. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2004), ACM, pp. 81–88.

19. YATANI, K., PARTRIDGE, K., BERN, M., AND NEWMAN, M. W. Escape: a target selection technique using visually-cued gestures. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2008), ACM, pp. 285–294.

20. YATANI, K., AND TRUONG, K. N. An evaluation of stylus-based text entry methods on handheld devices in stationary and mobile settings. In *MobileHCI '07: Proceedings of the 9th international conference on Human computer interaction with mobile devices and services* (New York, NY, USA, 2007), ACM, pp. 487–494.

21. ZHAI, S., AND KRISTENSSON, P.-O. Shorthand writing on stylus keyboard. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM, pp. 97–104.

QdQ: A New Keyboard for Writing Text on Small Soft Keyboards

Anders Houbak Kristiansen
Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
DK-9220 Aalborg East, Denmark
andershkristiansen@gmail.com

Frederik Larsen
Department of Computer Science
Aalborg University
Selma Lagerlöfs Vej 300
DK-9220 Aalborg East, Denmark
flabby@cs.aau.dk

ABSTRACT
This study evaluates the implementation (called QdQ) of a keyboard based on Directional Selection. It is compared against implementations of existing keyboards like Multitap and QWERTY with and without dictionaries using the results from a longitudinal user test.

Author Keywords
soft keyboards, text entry, touchscreen, QdQ, directional selection

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Input devices and strategies*

INTRODUCTION

Small mobile devices are used in many new areas where the desktop or laptop computer used to dominate. Devices like smartphones are used in new areas like browsing homepages, writing emails and participating in social networks. All of these new requirements require them to support the users in typing text with a decent efficiency in many different contexts. This is what this paper will try to improve, by introducing a new keyboard and evaluating it in a longitudinal user experiment.

One of the goals of this keyboard is to be independent on dictionaries, which has two important drawbacks we want to avoid. The first is, as [13] shows, that if only 15% of the words written is not in the dictionary all speed increase are lost. This can be words like abbreviations, usernames, passwords, email and web page address, which all are very important for the new areas the smartphones are used in. The other reason is the high cognitive load that the users feel when they are writing, because they need to focus on what the dictionary shows on the screen and on the next key they need to press [9].

Another goal is to avoid using a stylus. Many new smartphones, like the iPhone from Apple and the HTC Touch Diamond 2, uses a capacitive touchscreen that works by utilizing the small amount of electricity the human body conduct. This means they will not work with a standard stylus. Another reason for not using a stylus, is that the intense attention required can disrupt the users attention from the actual text input and it can feel tedious using a stylus over an extended period of time[19].

The keyboard developed in this paper is called QdQ (Quad direction QWERTY) and is based on Directional Selection (i.e. selecting something by dragging in an assigned direction) combined with a QWERTY keyboard. A screenshot of this keyboard can be seen in Figure 4.

Figure 1. The QdQ keyboard.

In this paper, we will compare this keyboard against several other keyboards based on existing text entry techniques, in a longitudinal user experiment. The other keyboards are a Multitap and a QWERTY keyboard with and without dictionary.

This paper deals with the following question: *How does QdQ compare to existing small soft keyboards?*

RELATED WORK

One way to design a keyboard for a small device where you do not have the space to place a full 26 keys keyboard (for the English language), is to have more characters on one button. A widely known example of this is Multitap, which is used on many mobile phones that are equipped 12 hard buttons. The alphabet a-z is distributed on buttons 2-9 as seen in Figure 2. For instance, to write `up`, you need to press two times on 8 and one time on 7. There is a timeout period usually 0.5-2 seconds, which is used to insert the selected character. For example, to write `on`, press three times on

Submitted for review to CHI 2010.

6, wait for timeout, and then press two times on 6. Some also implements a timeout kill button that a user can press to skip the timeout and insert the selected character. Because there is more letters on one button, the average keystroke per character (KSPC) is 2.0342[12] on the Multitap keyboard.

Figure 2. The Multitap keyboard.

Dictionaries are mainly used for two things, one is to gain disambiguation. A well known example of this is T9 by Tegic[5] that uses a Multitap layout. If 4, 3, 9 is pressed it will write `hey`, because this is the only possible word in the English dictionary that can be written from the 36 combinations that can be made. If it was 2, 6, 9 that was entered, it would write `box`, but if it was not the desired word, then there is a next word button which can be used to switch between the different words like `boy`, `bow` and `cow`. By using T9 it will decrease the KSPC of Multitap from 2.0342 to 1.0072[12] and thus gain a speed increase over Multitap. A way to gain disambiguation on a small QWERTY keyboard is to use a dictionary to correct errors in the output by looking at the surrounding characters at each key press, which is used on smartphones like Apple iPhone and on the Nokia Qt Extended platforms. The other thing a dictionary is used for is word prediction, which is used to get the KSCP below one. An example of this is [8], where wildcards can be used instead of writing the full word. Some of the problems with using dictionaries is if just 15%[13] of the entered words is not in the dictionary while using T9, all speed increases are lost. This prevents people from using abbreviations that are often used when messaging[13]. The problem impacts the performance of writing things like e-mail and homepage addresses.

Another way to write on a small mobile device is simply to not use buttons, but instead use gestures. An example of this is Shape Writing[19], where all words have their own gesture, which is the shape you get when drawing a line between each letter in the desired word on an image of a keyboard. Some gestures based techniques, like Graffiti by Palm[7], uses a novel character set that the user draws on the touchscreen to resemble handwriting. One of the disadvantages of most of these techniques is that they are made to be used with a pen and not the finger. This is not good, because the intense attention required when using a stylus can disrupt the users attention from the actual text input and can feel tedious when used over an extended period of time[19].

A technique, that focuses on a combination of high speed and accuracy when selecting small targets on a touchscreen rather than writing speed, is Escape[18]. It is used to select small targets which were occluded by the finger that was selecting. It works by giving all the objects a direction and to select an object the user use a simply dragging gesture in the direction of the object. This can be seen in Figure 3. One of the problems with Escape is that it does not work when many targets are close together because it then cannot map all of them to a few different directions, where it is still possible to select them. There is also a limitation on object density at the edge of the screen. For instance, it is not possible to drag down if the object is at bottom, but if you drag down just above it, it will still be selected if it is the nearest object with that gesture.

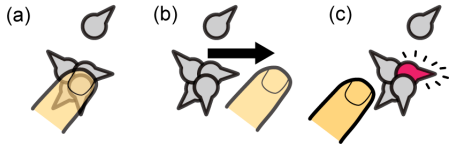


Figure 3. The Escape selection technique.

OUR KEYBOARD: QDQ

Our keyboard is a combination of a standard QWERTY keyboard and the target selection technique Escape[17]. This combination can help us achieve:

1. Independence of dictionaries, which would make it easier to write words that are not in the dictionary (like abbreviations, URL's, second language words and email addresses). As shown by [13], the dictionary actually *lowers* the writing performance if only 15% of the words are not in the dictionary. This is feasible due to a lower error rate.

2. Emulation of a larger keyboard, since the area in which you need to hit to select the desired key, is much larger than the keys, making it more feasible to use the fingers instead of a stylus.

3. Lower stress level, since the keyboards that are dependent on a dictionary or a stylus are known to have a high mental demand [9][19].

The keyboard, named QdQ, can be seen in Figure 4.

Figure 4. The QdQ keyboard in Arrow Mode.

Note that the layout is in Danish, which is because the test participants are from Denmark and should write in their primary language to avoid spelling trouble and other language related slowdowns. The SET button is used to switch between this and the other keyboards we have implemented for the evaluation, which are described in more details in the following sections.

As an example of usage, to write "hey", you need to follow this procedure:

1. Press down on letter `h`, move your finger a bit right and lift your finger again
2. Press down on letter `e`, move your finger a bit up and lift your finger again
3. Press down on letter `y`, move your finger a bit left and lift your finger again

It uses the position you press down to determine the center of the area to look for the letter and the position where you lift your finger to determine the direction. The area around the down-press is much larger than the actual keys, since the keyboard always chooses the closest key with the detected direction. Consequently, it actually emulates a larger keyboard, because you can press anywhere in that area and still get the desired key.

Regarding the performance of the keyboard, we can present the following hypotheses, based on the definition of the keyboard:

1. The WPM is higher than on a standard QWERTY keyboard, because the lower error rates makes up for the overhead of the drag.
2. The error rate is lower than on a standard QWERTY keyboard, because the area the user needs to hit is larger.
3. The workload is lower than on a keyboard with dictionary, because the dictionary adds to the stress level.

METHOD

To evaluate our technique, a comparative longitudinal user test using within-subjects was performed using the method described in this section. Each user did 20 sessions with five keyboards where they wrote both sentences and homepage addresses.

Participants

12 participants participated in this test but only 9 finished. Six out of the 9 participants were from Aalborg University, the last three participants were from outside the university. All 9 participants were volunteers and did not get paid for participating. Two of the participants where females and seven males, they had an age range from 21 to 49 (mean 28) years old. One of the participants was left-handed and none of the 9 participants were color-blind.

Six of the 9 participants use a mobile phone with a T9 keyboard with physical keys almost every day, where only two uses Multitap with physical keys and none of these participants had any experience when it comes to writing on a touchscreen based device. The last participant had some experience using a touchscreen based mini QWERTY keyboard on a smartphone. All of the participants were experienced when it comes to writing on a standard personal computer QWERTY keyboard.

Apparatus

The test was performed on two Openmoko Freerunner smartphones running Android with 128MB SDRAM and a 400MHz ARMv4T processor. It has a resistive touch screen with the dimension 480x640 pixels (1 pixel = 0.09mm), that does not support multi-touch, which implies writing with only one finger. The touchscreen was covered with a protective film that added more resistance when dragging than the standard touchscreen. The test program and keyboard was programmed in Java using the Android API over a period of 3 months. There was a high focus on little resource usage and a good performance, to ensure that it was not a bottleneck. A screenshot can be seen in Figure 5 with a QWERTY keyboard.

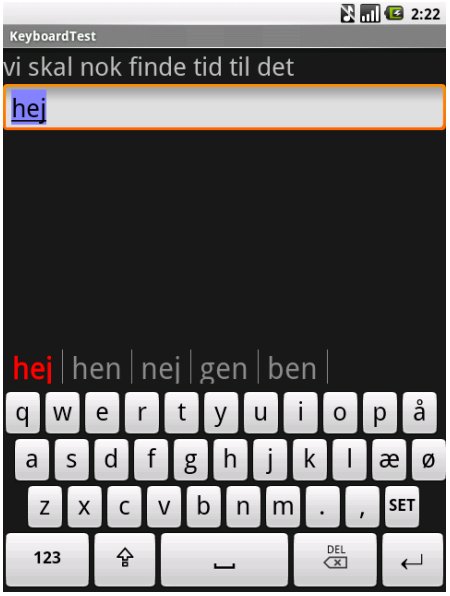


Figure 5. The QWERTY keyboard.

The test was performed with five different conditions, QWERTY keyboard, QWERTY keyboard with dictionary, Multitap, Multitap with dictionary and QdQ.

The QWERTY keyboard is a standard Danish alphabet QWERTY keyboard, which can be seen in Figure 5. The 123 button gives access to the numbers and symbols and the set button is used to change between the five conditions(will be disabled under the test, to prevent participants from accidently pressing it). The size of the buttons containing letters were 3.8x4.6mm, where the enter and shift buttons had a size of 6.5x5.8mm, 123 and delete buttons were a little larger with a size of 8.6x5.8mm and the space button with a size of 13.0x5.8mm.

The QWERTY keyboard with dictionary looks like the QWERTY keyboard, but utilizes a dictionary and learning to help the participant by correcting errors. The way the dictionary works is by looking at all adjacent keys within a 5.0mm radius. This was done to simulate a key with the size of 10x10mm. This was also tested to be the best size in a pilot study. The words, that the adjacent characters can create, is shown in a prioritized order in the view field (the bar above the keyboard in Figure5). They were prioritized using the shortest distance, from the coordinates where the user pressed down to the coordinates where they should have pressed down to get the word. This is then combined with learning that can make a word appear higher on the prioritized list after how many times it has been entered. When enter or space button is pressed the first word in the view field is selected followed by a space or enter. If the desired word is not the first, it can be selected by pressing it, and only that word is inserted. The view field can also scroll horizontally if there are more words than there is room for on the screen.

The Multitap keyboard was implemented such that all of the buttons on the Multitap keyboard had the size of 13.0x8.6mm which was the maximum size possible, where the test program still could be used. The timeout were chosen to be one second to resemble a real world implementation of Multitap on a Sony Ericsson W800i. We decided not to implement a timeout kill button because just pressing the first character in the view field, which rotates at every key press, has the same function. It can be seen in Figure 6

Figure 6. The Multitap keyboard.

The Multitap keyboard with dictionary was implemented to resemble T9[5], which is a keyboards with a Multitap layout that is using dictionary. It also uses learning to decide the order of the found words. The view field works exactly like in the QWERTY keyboard with dictionary.

The QdQ keyboard is the implementation of our technique, where the button layout (not the look) and size is the same as in the QWERTY keyboard. A minimum drag distance of 1.35mm was chosen from a pilot study as the distance, where a drag was long enough to not just be a simple click. The only way to enter letters was by performing a drag and not by clicking. The other buttons (bottom row) still works like in the original QWERTY keyboard, where they need to be clicked. This was done to prevent the inconvenience it would be to drag down to the bottom of the screen, because of the edge. Additionally, the pilot study did show that there was no problem doing it this way.

We will use the following short names to refer to the keyboards:

- QWERTY: The standard QWERTY keyboard.
- QWERTY (dic): The standard QWERTY keyboard with dictionary.
- Multitap: The Multitap keyboard
- Multitap (dic): The multitap keyboard with dictionary.
- QdQ: The QdQ keyboard using arrows for mapping to direction.

Experimental Design

Within-subjects were used for the test with 20 sessions as factor crossed with the five conditions QWERTY keyboard, QWERTY keyboard with dictionary, Multitap, Multitap with dictionary and QdQ. There were six conditions in the test where the last one was a variation of QdQ but only five was used in this article. The dependent variables were entry rate (Words per Minutes) and accuracy (Error rates).

To prevent carryover effects, the keyboards were distributed in a 6x6 balanced roman square where participant one started with the first row, then second row next session. Participant two started with the second row and so on for all the participants. This was done to mix the order of the keyboards. All of the sentences were picked randomly (but distinct for the specific sessions) from a set of 602 sentences. They only contained lower case characters. All of the sentences were obtained from three different short stories[4][6][3]. The sentences had a length between 16-44 characters (with an average of 28.5), consisted of 3192 words (1084 unique words) and the average word length were 4.6 characters. The homepage addresses were randomly selected from the 100 most popular homepages in the world[1] without the ”http://www.”. They contained lower case characters, numbers, ”-” and ”.”. The order of the eight sentences and the three homepage addresses changed from keyboard to keyboard and also from session to session by alternating between them. The words in the dictionary was from [2] where all the words from the sentences were added this was done to enable us to look at the data from the sentence as 100% represented in the dictionary. None of the homepages addresses was in the dictionary to give us opposite data, 0% representation. It was decided that every participant started with their own empty learning which they kept building on throughout all of the sessions. This was decided to be the most fair, and resemble the real world the most, compared to no learning or perfect learning.

Procedure

Before the first session the participants got the chance to write a few sentences with all of the different keyboards and

afterwards ask questions if there was something they did not understand. They were instructed to write as quickly and accurately as possible, where they were allowed to have uncorrected errors. Sentences were instructed to be read meticulously before stating to write, to keep FOA (focus of attention) down. The sentences remained visible under the test to prevent spelling errors and prevent them from forgetting the sentences. In each of the sessions they had to write eight sentences and three homepage addresses with each of the keyboards. After they were finished with one keyboard they had to change the keyboard on their own and continue the session until they were done, with all of the keyboards. There were no restrictions on where they were located under the test, but they were asked to make sure they sat comfortably and undisturbed before starting. Every time a participant was finished with a couple of sessions using the smartphone they had to bring it to us so we could take a backup and transfer test data from another participant to it. This was a necessary such the two smartphones should be used by the seven participant in the same time frame. They also had the opportunity to borrow one of the smartphones home at the evening or at the weekend to test. Seven of participants had a deadline of 20 days after the test began to complete the sessions however this timeframe had to be extended to 30 days because only one was finished at the deadline. The last three participants got an seven days deadline to complete all the sessions. How the sessions were distributed was up to the individual participant. Pauses could be held between two sentences, between the different keyboards and at the end of each session. The last session was performed in a usability lab with the participant walking on a treadmill with a speed warring from 1.5-3.5 Km/h as recommended in [14], to see how the keyboards would perform if the participant was moving. After all the sessions the participants took a NASA TLX test even if is some discrepancies[15].

Data Collection

For each key pressed, we collected a time stamp which started from when the first key was pressed in a sentence, key code (including space and backspace), session number, sentence number, keyboard used and a reference to the written sentence which was used to help find all the uncorrected errors. All test data was automatically collected on the mobile phone and saved to a mini SD Card where all of the participants data were stored separately.

Data Analysis

The data that was collected was used to find learning effects, WPM and error rates. WPM is calculated as the time used to type the sentence plus the average time to correct an error multiplied with the number of uncorrected errors in the sentence. The combined time is divided with the number of characters in the sentence including spaces minuses the first character. This time was then divided by 60 to get it in minutes. Finally this is divided with five as the accepted word length to get WPM.

To measure the error rates, we use corrected, uncorrected and total error rate as described in [16]. We define CPS as Character per Second, without any error calculations included. For statistical analysis, we use the F-test throughout this paper.

RESULTS

In this section, we will show the results from the test which relates to answering the research question. Additionally, we will prove or disprove the hypotheses presented previously in this paper.

Entry Rates

In Figure 7 all the keyboards are compared in terms of WPM when entering sentences. It shows that QWERTY, QWERTY (dic) and Multitap (dic) are significantly faster than QdQ and Multitap (p ranging from $p = 0.0027$ to $p = 0.0002$). QdQ is also significantly faster than Multitap ($F_{1,16} = 26.71, p = 0.0009$). All the keyboards had a learning effect and have seen a speed increase over the 19 sessions ranging from 46.6% for Multitap to 80.2% for Multitap (dic).

Figure 8 shows the learning curves (approximated extrapolation) for the data from Figure 7, which shows that even after a potential 50 sessions, the order of the keyboards are still almost the same, except that QWERTY is slightly above QWERTY (dic). The specific information about these learning curves is shown in the table below (where the WPM learning curve is defined as a $a * x^b$ function and R^2 is the correlation coefficient):

	a	b	R^2
QWERTY	12.96	0.14	0.7
QWERTY (dic)	12.62	0.17	0.83
Multitap (dic)	11.29	0.2	0.75
QdQ	9.97	0.16	0.71
Multitap	7.68	0.13	0.7

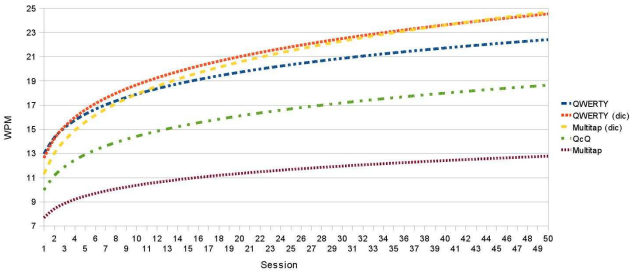


Figure 8. Trend: WPM by keyboard and session (sentences)

In Figure 9 all the keyboards are compared in terms of WPM when entering homepage addresses. The only significant differences between entering sentences and homepage addresses is for Multitap (dic) ($F_{1,16} = 21.58, p = 0.0003$) and QWERTY (dic) ($F_{1,16} = 6.49, p = 0.0215$). Typing sentences is 236% faster with Multitap (dic) and 48% faster with QWERTY (dic).

Hypothesis 1: The WPM is higher than on a standard QWERTY keyboard, because the lower error rates makes up for the overhead of the drag

Answer: This is false, even though Arrow does have a lower error rate.

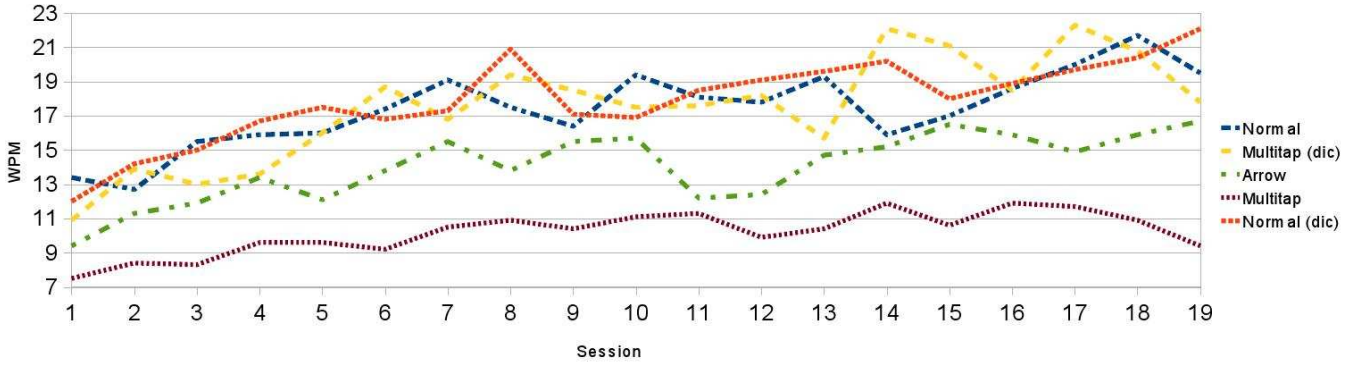


Figure 7. WPM by keyboard (sentences)

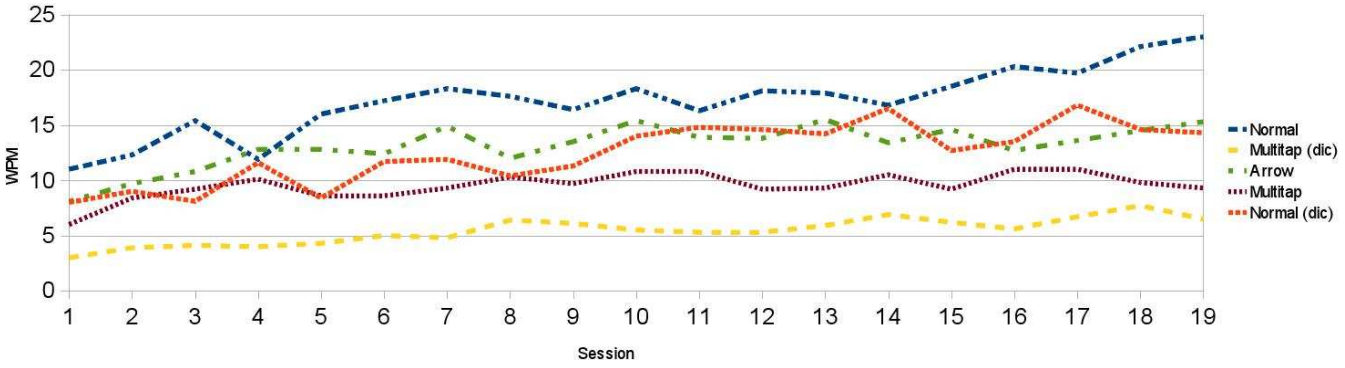


Figure 9. WPM by keyboard (homepage addresses)

Accuracy

The total error rates by keyboard for sentences and homepage addresses can be seen in Figure 10 and Figure 11. Both figures shows the improvement of the error rate over the 19 sessions for all the keyboards. For sentences, QWERTY had a significantly higher error rate than Arrow ($F_{1,16} = 6.76, p = 0.0316$), QWERTY (dic) ($F_{1,16} = 8.24, p = 0.0208$) and Multitap (dic) ($F_{1,16} = 6.48, p = 0.0344$). For homepage addresses, Arrow had significantly lower error rate than both QWERTY (dic) ($F_{1,16} = 10.77, p = 0.0112$) and Multitap (dic) ($F_{1,16} = 36.56, p = 0.0003$).

The only significant differences between entering sentences and homepage addresses is for Multitap with dictionary ($F_{1,16} = 33.69, p = 0.0004$) and QWERTY with dictionary ($F_{1,16} = 18.66, p = 0.0025$). Since we have tested both of the keyboards without a dictionary, where there were no significant results, we can conclude that keyboards with dictionary causes a significantly higher error rate when entering words that are not in the dictionary.

Hypothesis 2: The error rate is lower than on a standard QWERTY keyboard, because the area the user needs to hit is larger

Answer: This is true, because Arrow does have a lower error rate than QWERTY for both sentences and homepage addresses.

Walking vs. Sitting

The participants were asked to walk while doing session 20. Figure 12 shows an overview of how fast the keyboards were while walking and sitting. The WPM used for sitting in this graph is defined as being the average WPM with the given keyboard for session 17-19. Sitting is defined this way, because session 19 alone (as shown in Figure 7) does not provide good numbers to test against due to the low WPM in session 19. The numbers used for walking was only the WPM for session 20. From the comparison in Figure 12, we can see that all keyboards show a drop in WPM when writing sentences while walking. This drop is significant for QWERTY ($F_{1,16} = 5.71, p = 0.0439$) and QWERTY (dic) ($F_{1,16} = 16.69, p = 0.0035$). If we look at homepage addresses we can see the same trend.

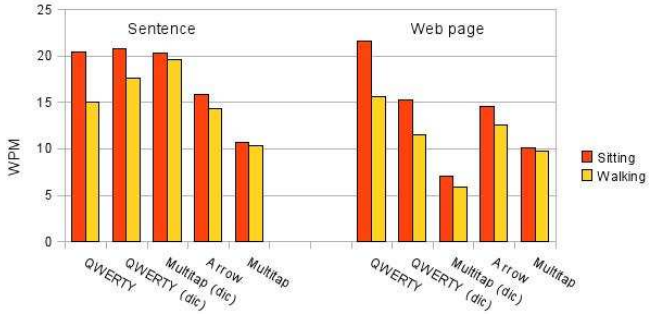


Figure 12. Comparing Sitting and Walking (WPM)

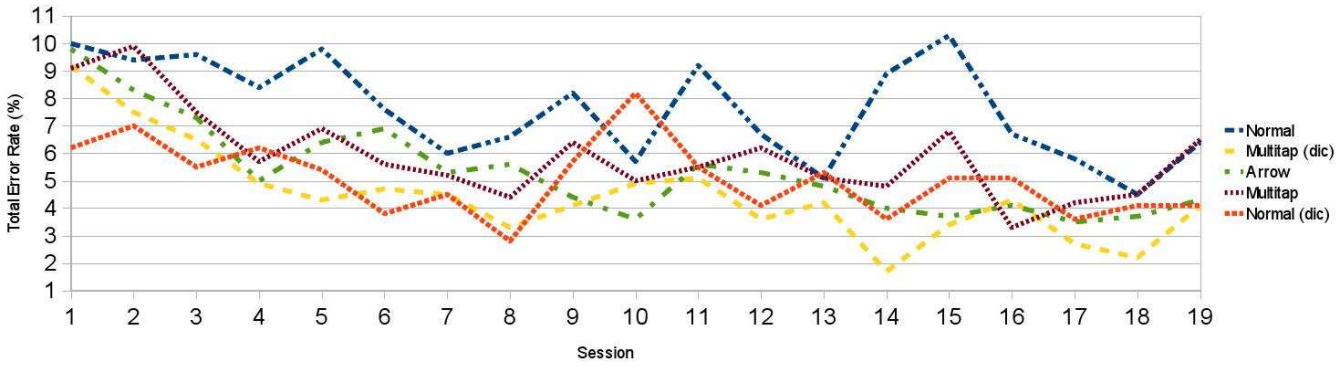


Figure 10. Errors by keyboard (sentences)

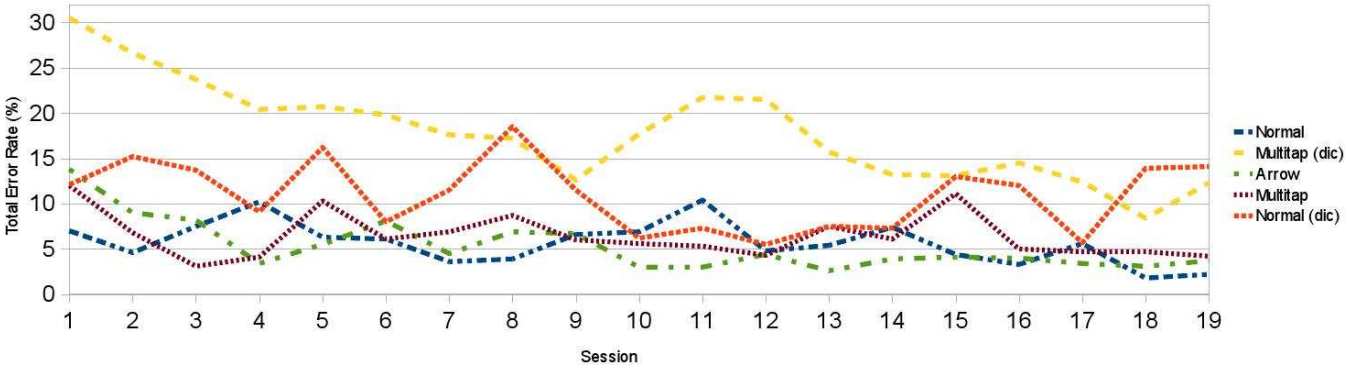


Figure 11. Errors by keyboard (homepage addresses)

Figure 13 shows an overview of the total error rates of the keyboards while walking and sitting. From this comparison, we can see that all keyboards had a higher total error rate when typing while walking. On average the error rates were 72.6% higher when walking (from 5% to 8.7%). When looking at sentences the Arrow keyboard was comparable to QWERTY (dic). For homepage addresses, Arrow was the keyboard with the lowest error rate while walking and at the same level as QWERTY when sitting.

The results showing a significant difference between entering sentences and homepage addresses for dictionary based keyboards are also clearly visible in this figure. They also had a 255.2% higher error rate when walking and writing homepage addresses compared to walking and writing sentences. The keyboards without a dictionary were staying at about the same level.

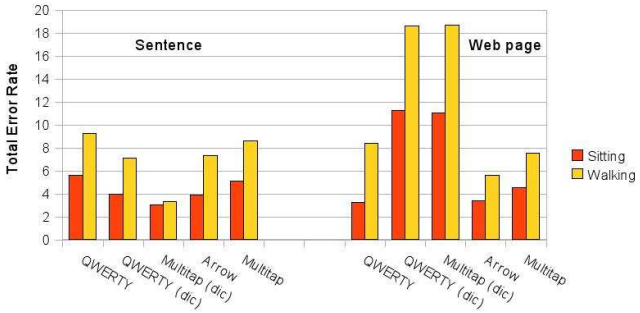


Figure 13. Comparing Sitting and Walking (Errors)

NASA TLX

The results from the NASA TLX test can be seen in the following table. It was done to see how the participants felt the workload for Directional Selection, compared to what they were used to.

	Arrow workload	Usual workload	Usual keyboard
Partitipant 1	54	73	QWERTY
Partitipant 2	80	54	T9
Partitipant 3	39	24	T9
Partitipant 4	88	19	T9
Partitipant 5	32	49	Multitap
Partitipant 6	46	63	T9
Partitipant 7	52	49	T9
Partitipant 8	34	33	T9
Partitipant 9	75	66	Multitap
Average	**56**	**48**	-

The average workload for the keyboard they usually used

were 48, which for Arrow is 56. There is no significant difference ($F_{1,16} = 0.71, p = 0.4244$) between these, but even if there were, we would have to take their much longer experience with their usual keyboard into account.

Hypothesis 3: The workload is lower than on a keyboard with dictionary, because the dictionary adds to the stress level

Answer: The numbers shows that Arrow might have a higher workload than T9, but since the difference is not significant, we cannot say for sure.

DISCUSSION

We chose to compare QdQ to Multitap, because it is used as a baseline in many studies. Our implementation of Multitap had an entry rate of 13 WPM which is somewhat lower than the 15.5WPM in [13]. This difference is reasonable since none of the participants had used the view field as a timeout kill button and because we used soft buttons, rather than hard buttons as they did.

Multitap with dictionary was chosen so that there was a keyboard with a dictionary to compare against keyboards without any dictionary and because many people are familiar with and using T9 daily. In our test it had an entry rate of just below 24 WPM. In [10] they get 15.1 WPM when using newspaper text type and 25.7WPM with a chat text type. Since our text type is a lot closer to their newspaper text type than chat, it is a little fast, which is to expect since six of the nine participants used a T9 keyboard daily.

The QWERTY keyboard was added to get some data from participants use of a small QWERTY keyboard only using their fingers to write. Additionally, we wanted see how the basic keyboard would perform against keyboards that tries to help people to write more correct and faster like QdQ and QWERTY keyboard with dictionary. The experiment we could find that resembles our QWERTY keyboard the most in size, and that it was performed on a touchscreen, was [14] with 22 WPM, but the participants were using a stylus. 22 WPM is somewhat low when compared to the QWERTY keyboards 22WPM entry rate, since their participants are testing using a stylus and ours are testing using a finger. If we try to take a look at the error rate, we have to look at [11] or at articles with focus on selection instead of keyboards because there is not a lot of research in the field of writing using a finger on a touchscreen. So we have looked at [11] which shows an error rate of 11% when using a button size of 7.2x6.5mm. Another article[17] shows 46% error rate when using the tip of a finger on a touchscreen and 30% with the nail when using 5.2x5.2mm buttons. Our letter buttons are 3.8x4.6mm which is 62.6% smaller than in [11] and 35.4% than in [17]. So there is to expect an error rate higher than 30% at least, but we did only see 6-10% which also explains why the WPM is so high.

Both [11] and [17] used automatic data collection so they could not have caught any more errors than we could. If we had video or was observing them, we could have caught errors like when a participant did not press hard enough on the touchscreen. In both articles they also instructed their participants to be as quickly and accurately as possible as we did. [11] sees the buttons as including the gap between the buttons which we also do, so there is no difference there. They also include the same errors as we do in the total error rate, so that is not a contribution to any difference in the error rate. [17] only has one button on the screen that the user has to select to proceed or press a next trial button if they give up. To see if it has an effect, we could try to look at some larger buttons (7.8x7.8mm) in [17]. They get a 9% error rate using nails and 18% with the tip of the finger. Because only 4 of the 13 participants in [11] are using their nail, it shows us that [17] have a higher error rate than [11], but this is a small difference and nothing in the area we are seeing.

The differences in error rate were not a product of participants using stylus in our experiment, since we got the same data from sessions where the participant were observed by us and from a small test where we tried to complete a session with and without a stylus ourselves. With stylus we saw a 1.8% error rate and by using a finger we saw 7% which confirms our experiment. The average size of the index finger of our participants were XXmm which is close to the anthropomorphic average at 18.2mm[11], and one participant even used her thumb. There is no information about the finger size of the participants in [11] or [17], so we cannot say if it had any effect. From what we can see there is no good explanation, as to why there is such a big difference from our error rate to the error rate reported in [11] and [17]. This is something that should be investigated further in a new study.

We noticed that some of the participant data became irregular. We traced this to the fact that there was no time restriction on how much time there can be between two sessions. Their learning curve would go up and then drop down after they did some sessions. This happened if they had a long break, like one participant who had a 14 days break between two sessions. This effect was also clearly visible if they performed many sessions in a row. The worst problem with this was that a single users data can become very irregular which leads to that the error deviation in the graphs was high. But they did still learn throughout all the sessions and all participants got a learning effect on all the keyboards over the 20 sessions and they got some good entry rates compared to other experiments. This can be seen in Figure 14, that shows the learning curve and error deviation for all the keyboards in one graph. Even on Multitap with dictionary we saw a learning effect which is a keyboard type seven out of 10 participants uses daily. This tells us that for a longitudinal test the user must have some restrictions on how long time they can work in a row, such that they does not get exhausted and therefore loose performance.

Some of the participants commented that they felt that they had not used QdQ enough for it to be efficient, and if they had the chance, it could be faster than the QWERTY keyboard in mixed use. One also said, that she had many errors in QdQ, when she did more than one session right after another, which was not something she felt with the others.

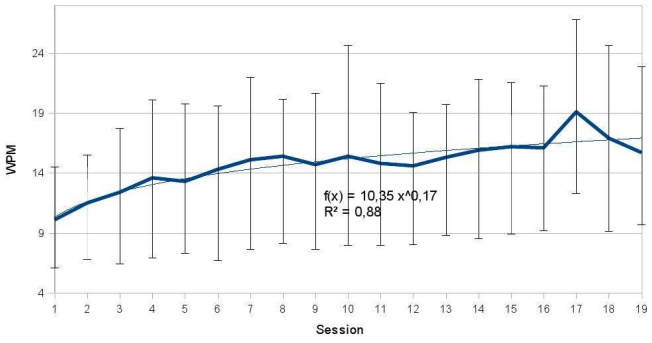


Figure 14. WPM by session (sentences)

CONCLUSION

In this article we attempted to answer the following question: *How does QdQ compare to existing small soft keyboards?*

The simple answer to this question is that it did perform about average of the other keyboards in the test when it comes to sentences and the participants were sitting down. When replacing the sentences with homepage addresses, the Arrow keyboard was above average, especially because the Multitap keyboard with dictionary and the QWERTY keyboard with dictionary suffered from the text not being in the dictionary.

Asking the participants to walk did not affect the WPM much, but the error rates were a lot higher, except for Arrow (sentences and homepage addresses), Multitap (sentences and homepage addresses) and Multitap with dictionary (sentences only). This tells us that even though the Arrow keyboard generally is a bit slower, it is much more resistant to outside interruptions than the other two QWERTY based keyboards.

As answer to the research question, the Arrow keyboard performed reasonable, but there are still improvements to be made. Generally speaking the Arrow keyboard is versatile when it comes to changes in environment or the type of text, while still maintaining a reasonable entry speed.

Limitations

The following limitations should be considere when using the results from this paper:

- The surface of the screen were a bit resistant when it comes to dragging compared to the default screen, which might cause a lower performance when dragging.
- The screen does only support single-touch, where a multi-touch is becoming more and more popular. On a multi-touch screen the numbers could be very different.
- The participants were not instructed in how long a break they should have between sessions. Too long breaks did lower the performance because they got out of the routine, and too short did also lower the performance, because they got tired. This is a result from the experiment.
- We did only have one walking session, and based on the difference between the sessions before that one, the results may not be accurate.

Future Work

QdQ can be improved in future implementations, by using another layout where the mapping is more obvious. This could be one like Multitap, where we exploit the large buttons, by placing the letters such that their position on the button, relative to the middle, symbolizes the dragging direction to get that letter. The point is that the technique has proved feasible, but to gain the performance necessary to replace existing techniques, it should be attempted implemented on other layouts that QWERTY.

REFERENCES

1. Alexa the web information company. `http://www.alexa.com/`.
2. Danish wordlists. `ftp://ftp.mathematik.uni-marburg.de/pub/mirror/openwall/wordlists/languages/Danish/`.
3. Rejse mod vinter. `http://www.fyldepennen.dk/tekster/29289`.
4. Sort og hvidt hjerte. `http://www.fyldepennen.dk/tekster/37687`.
5. Tegic. `http://www.nuance.com`.
6. Uden titel. `http://www.fyldepennen.dk/tekster/38526`.
7. CASTELLUCCI, S. J., AND MACKENZIE, I. S. Graffiti vs. unistrokes: an empirical comparison. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2008), ACM, pp. 305–308.
8. CHURCH, K., AND THIESSON, B. The wild thing! In *ACL '05: Proceedings of the ACL 2005 on Interactive poster and demonstration sessions* (Morristown, NJ, USA, 2005), Association for Computational Linguistics, pp. 93–96.
9. COCKBURN, A., AND SIRESENA, A. Evaluating mobile text entry with the fastap keypad. In *HCI 2003* (2003), British HCI Group, pp. 77–80.
10. JAMES, C. L., AND REISCHEL, K. M. Text input for mobile devices: comparing model prediction to actual performance. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2001), ACM, pp. 365–371.
11. LEE, S., AND ZHAI, S. The performance of touch screen soft buttons. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems* (New York, NY, USA, 2009), ACM, pp. 309–318.
12. MACKENZIE, I. S. KSPC (keystrokes per character) as a characteristic of text entry techniques. In *Human Computer Interaction with Mobile Devices* (2002),

vol. 2411 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 405–416.

13. MACKENZIE, I. S., KOBER, H., SMITH, D., JONES, T., AND SKEPNER, E. Letterwise: prefix-based disambiguation for mobile text input. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2001), ACM, pp. 111–120.

14. MIZOBUCHI, S., CHIGNELL, M., AND NEWTON, D. Mobile text entry: relationship between walking speed and text input task difficulty. In *MobileHCI '05: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services* (New York, NY, USA, 2005), ACM, pp. 122–128.

15. PARK, S., HARADA, A., AND IGARASHI, H. Influences of personal preference on product usability. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 87–92.

16. SOUKOREFF, R. W., AND MACKENZIE, I. S. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM, pp. 113–120.

17. VOGEL, D., AND BAUDISCH, P. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2007), ACM, pp. 657–666.

18. YATANI, K., PARTRIDGE, K., BERN, M., AND NEWMAN, M. W. Escape: a target selection technique using visually-cued gestures. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2008), ACM, pp. 285–294.

19. ZHAI, S., AND KRISTENSSON, P.-O. Shorthand writing on stylus keyboard. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2003), ACM, pp. 97–104.