TwoStick: Writing with a Game Controller

Thomas Költringer[♦], Poika Isokoski[ξ], Thomas Grechenig[♦]

[♦]Vienna University of Technology
Research Group for Industrial Software (INSO)
Wiedner Hauptstrasse 76/2/2, 1040, Vienna, Austria
{thomas.koeltringer, thomas.grechenig}@inso.tuwien.ac.at

[ξ]University of Tampere
TAUCHI, Department of Computer Sciences
FIN-33014, University of Tampere, Finland
poika@cs.uta.fi

ABSTRACT

We report the design and evaluation of a novel game controller text entry method called TwoStick. The design is based on the review of previous work and several rounds of pilot testing. We compared user performance with TwoStick experimentally to a selection keyboard which is the de facto standard of game controller text entry. Eight participants completed 20 fifteen-minute sessions with both text entry methods. In the beginning TwoStick was slower (4.3 wpm, uncorrected error rate = 0.68%) than the selection keyboard (5.6 wpm, 0.85%). During the last session TwoStick was faster (14.9 wpm, 0.86% vs. 12.9 wpm, 0.27%). Qualitative results indicated that TwoStick was more fun and easier to use than the selection keyboard.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces - Evaluation/Methodology, Input Devices and Strategies.

Keywords: Text entry, game controller, gamepad, joystick, selection keyboard, Quikwriting, TwoStick

1 INTRODUCTION

In the early 20th century the French aviator Robert Esnault-Pelterie invented the joystick [24]. Since then, translation of human will into machine movement via a stick has become widely adopted. Nowadays airplanes, ships, mobile phones, game avatars, and many other things are operated with joysticks.

The use of joysticks in computer games began early on. The Atari 2600 from 1977 was an early gaming product with a joystick. A joystick has been a standard part of game controllers at least since Nintendo N64 launched in 1996. With the exception of Nintendo Wii, the controllers shipped with popular game consoles have designs similar to the Microsoft Xbox 360 controller shown in Figure 1.

Game consoles are also used for other tasks besides gaming. In some cases they are the main entertainment appliances in the living room. Users of game consoles can access pictures, music and movies. The consoles can also be connected to the Internet for playing against remote opponents, for messaging, or for browsing the web. Consequently, the game controller has become involved in tasks for which it was not designed.

Most text entry methods for game consoles are adaptations of methods that were originally developed for different input devices. Maybe writing with a gamepad would be easier if we had a text entry method that was designed for gamepad use.

In this paper we report our search for text entry methods for gamepads. We were looking for methods that are *fast to use* and *easy to learn*. First we summarize and discuss previous work. Then we describe TwoStick, a new text entry method for gamepads. After that, we report the results of an experiment where we compare a gamepad operated QWERTY selection keyboard and TwoStick.

Graphics Interface Conference 2007
28-30 May, Montréal, Canada

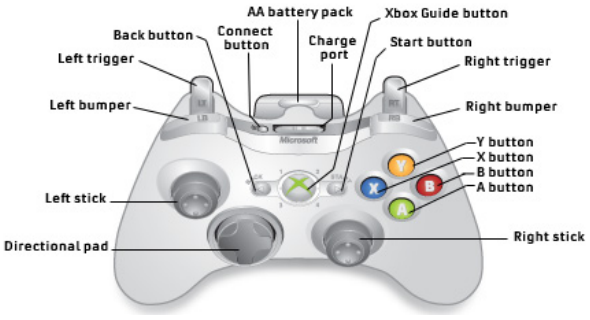


Figure 1. Sketch of a Microsoft Xbox 360 game controller.

2 GAME CONTROLLER TEXT ENTRY

In the beginning of computer gaming there were arcade games. In 1979 the game Asteroids introduced the feature to label the user's score in a high score list allowing players to compete for the highest score. For entering the label the games used the date stamp technique [11, 23] where the user scrolls through characters and presses a button when the proper character is highlighted.

Nowadays, the typical text entry method for gamepads is an onscreen selection keyboard [20, 23]. Usually the QWERTY or alphabetic key layout is used. A selection keyboard is used by moving the highlighted focus on the desired key and then selecting that key. Usually a mini-stick or a directional pad (D-pad) is used for movement, and a button for selection. Sometimes most frequent characters are assigned to buttons to provide shortcuts for entering them. Selection keyboards can be cumbersome, because large distances have to be traversed between some keys.

Recently Sony has shipped gaming devices (PlayStation 3, PlayStation Portable) with a selection keyboard version of the multi-tap text entry method used in mobile phones. Having fewer keys reduces the distances to be traveled between the keys. The cost of this reduction is the ambiguity arising from several characters being associated with each key. Nintendo with the Wii is shipping both, a QWERTY selection keyboard and a multi-tap keyboard, letting the user choose the preferred method. According to press releases [26] Sony and Nintendo are planning to add a dictionary-based disambiguation technique to their systems for faster text entry.

Wilson and Agrawala [20] proposed a partial solution to the keyboard traversal problem by splitting the keyboard in two. Each half had its dedicated cursor operated by its own mini-stick. Their experiment showed that this solution was faster for novice users than a non-split layout. They also found the QWERTY selection keyboard layout to be faster than an alphabetic layout.

Text entry methods are provided by the operating system of the game console. However, game producers can still implement whatever they want within the games. Consequently, innovative text entry methods have been implemented in games. For

Method	Performance: Novices	Training	Experts	Source
Quikwriting	4 wpm	5 hours	13 wpm	[8]
Date Stamp	4.4-9.1 wpm	-	-	[23,11]
MDITIM	5.6 wpm	-	-	[5]
ABC skb	5.8 wpm	-	-	[20]
QWERTY skb	6.2- 6.5wpm	-	-	[23,20]
EdgeWrite	6.4 wpm	unknown	11-14.7	[22,23]
Dual QWERTY	7.1 wpm	-	-	[20]

Table 1. Performance of game controller compatible text entry methods.

example, in the game Rainbow Six 3, characters are shown on a circular menu and entered by deflecting the mini-stick to the direction of a character and pressing a button. Unfortunately mapping at minimum 28 characters (alphabet, space, backspace) to a circular menu, results in narrow 12.9 degree menu slices. This kind of accuracy is hard to achieve with a mini-stick. Proschowsky et al. [15] had a similar problem when mapping a keyboard to a click wheel (such as the wheel used in Apple iPods). They alleviated the problem by supporting selection with a predictive algorithm.

Isokoski and Raisamo [5] developed a gesture based device independent text input method called MDITIM. The idea was to design a gesture alphabet that could be used on different input devices, including joysticks. The authors speculated that the tactile feedback from the frame around a stick helps in gesture input.

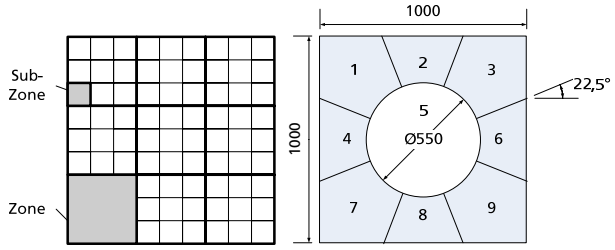
Wobbrock et al. [23] developed a gesture alphabet called EdgeWrite that explicitly uses the edges of the stick movement as input. Wobbrock et al. compared novice performance with date stamp, selection keyboard, and EdgeWrite using one mini-stick on a gamepad. After some training participants were faster with EdgeWrite than with the other methods. Novice input with the selection keyboard was nearly as fast as EdgeWrite. Users felt that the keyboard was the slowest and most frustrating of all tested methods. Edgewrite has been implemented on a multitude of input devices. Using it is possible at least with keys, joysticks, styli and trackballs.

Quikwriting [14] was originally developed for stylus use. Isokoski and Raisamo [6] adapted Quikwriting for joystick use. At the end of a 20-session experiment the participants achieved the rate of 13 words per minute (wpm) with the mini-sticks on a game controller (see Table 1). Isokoski and Raisamo concluded that Quikwriting is not easy to learn, but with adequate training text entry speed increases.

Microsoft XNav [7] is an adaptation of Quikwriting with an alphabetical character layout. Whereas Quikwriting implementations are mainly research prototypes, XNav is a more deliberate implementation for industrial use. Perlin, the original developer of Quikwriting has also presented ideas on how to modify Quikwriting to better suit gamepad input [13]. It seems that if the project had continued to a gamepad operated prototype, the results might have been similar to our solution in this paper. Neither Microsoft, nor Perlin have published results on the performance of gamepad based Quikwriting derivatives.

Table 1 shows data on the performance of joystick-operated text entry methods. The data for date stamp in [11] was measured using keys, but we would expect similar performance with a stick. The 14.7 wpm figure for joystick-operated EdgeWrite was achieved by Wobbrock himself [22] after unknown amount of training. The fastest participant in the Quikwriting experiment by Isokoski and Raisamo achieved the rate of 17.8 wpm.

Other text entry systems such as T-Cube [18], KeyStick [8] and Weegie [9] have been, or could easily be, adapted to joystick use.



a) The basic layout is divided into zones and sub-zones

b) Dimensions and zones of the joysticks input area

Figure 2. TwoStick visualization and input plane

Weegie uses two sticks to operate two different layouts which may improve writing speed [20]. myText [2] and the MobileQWERTY [12] are other commercially available methods for joystick based text entry.

3 TwoStick

TwoStick is a novel text entry method for game controllers. The idea of TwoStick was derived from Quikwriting [13, 14] and the visual feedback resembles that of TNT [3]. The original developers of Quikwriting [14] and others [6, 7] have noted that using a stylus for Quikwriting is just one of the possibilities. Quikwriting divides the input area into nine zones that are mapped to nine tokens. Sequences of these tokens are translated to characters. TwoStick operates with the same main concepts.

3.1 User's View

In TwoStick the user is presented a character layout on a two level grid (see Figure 2a). The grid consists of 9 zones each of which consists of 9 sub-zones. In total there are 9x9=81 square shaped sub-zones. Each sub-zone is associated with a character that can be entered. The 9 zones are mapped to the 8 directions and the center zone of a joystick (see Figure 2b). The first level zone selection is mapped to one stick, and the second level character selection is mapped to the other stick.

Figure 3 gives the example entering 't' with TwoStick. In the beginning both sticks are centered. Then, the lower-left zone (zone 7) is selected with the zone selection stick. After this, the character ('t' = sub-zone 2) is selected with the character selection stick. The entry of the character happens when the character selection stick returns to the center. As shown in Figure 3, the highlighting on the display follows the stick movements making the operation easy to follow.

Note that if the character selection stick has not been centered before the zone selection stick is moved, the character selection will retain its relative position. I.e. if a user has highlighted 't' as in Figure 3c and then moves the zone selection stick over zone 1, nothing will be entered. Instead 'b' will be selected. This makes it possible to correct zone selection errors before entering the character but also makes it possible to make errors if the zone selection stick is moved too soon.

This kind of interaction makes it inconvenient to have characters at the centers of the first level zones. This results in maximum number of 72 characters on the layout. Some of these, as well as buttons on the gamepad can be used for mode changing. Thus, the maximum number of characters is unlimited in theory. However, accessing other than the 72 characters on the default layout is difficult and time consuming.

To be able to place 81 characters on the layout we have experimented with a version of TwoStick where characters are selected as described above, but entered by pressing a button. We

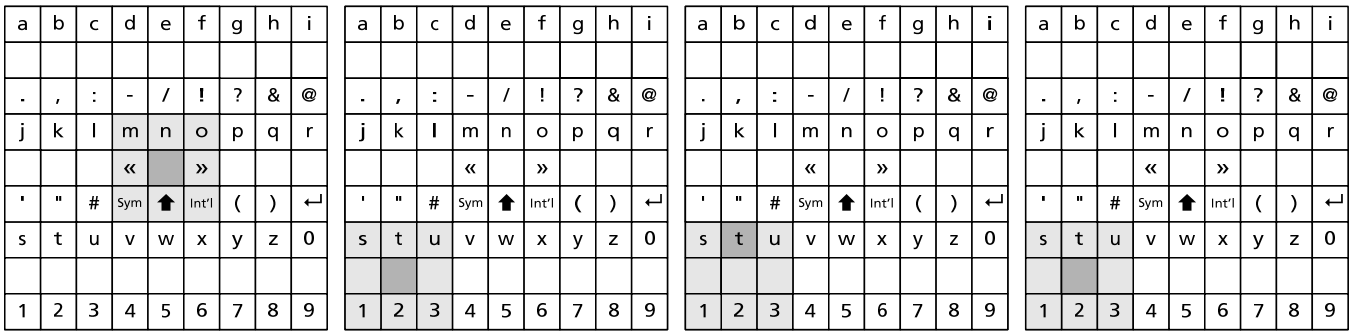


Figure 3. Entering ‘t’ with TwoStick

expected to gain more accuracy in entering characters. Informal tests showed that users preferred the design without the need to press the button. Consequently the design without button pressing was selected for the experiment reported later in this paper.

3.2 Character Layout

An optimal layout for TwoStick would involve placing the most frequent characters within one zone to reduce the amount of zone changes. Such an optimization would result in a layout that is more difficult to learn than an alphabetic ordering of the characters. Walk-up usability is very important in text entry methods. Consequently we chose an alphabetic layout.

The initial idea for the character layout was to arrange the characters in the periphery of the input area as in Quikwriting [14] (see Figure 4). This design would obviously waste some space. Also, tests during the design phase showed that users preferred upwards movements of the character selection stick sticks. We concluded that it might be a good idea to place the alphabet on the upper sub-zones (1, 2, 3) and the less frequently used characters on the bottom (7, 8, 9). Following these decisions yielded most of the layout shown in Figure 3.

For the shifting we decided to use the commonly used auto-backshift mechanism. This means that when the first character from the shifted layout is selected, the default layout automatically shifts back. When a persistent shift is needed, the user needs to enter the shift character twice. Returning from a persistent shift happens by entering the shift character one more time. The shift functionality for accessing other layouts was placed in the bottom row of the central zone so that it would be fast to use. Three layouts were used in addition to the default layout seen in Figure 3. These are the upper case characters (arrow up), symbols (Sym) and the international characters (Int’l).

Space can be entered in the central zone by selecting the sub-zone on the right and backspace by selecting the sub-zone on the left. After placing the most frequent non-alphabetic characters and numbers on the main layout we still had some empty cells. This space could be used for internationalization. For example, the language-specific alphabet of many European languages would fit in the remaining space. For our English layout we decided to add space and backspace in every first level zone. These two characters are needed very frequently in text entry. Having them available without moving the zone selection stick should improve performance and comfort of use. The symbols for space and backspace were not shown outside the central zone to avoid clutter, but the functionality was there.

3.3 Implementation Details

The dimensions of the areas shown in Figure 2b are a result of an iterative design and testing cycle. More formal experiments could help to further fine-tune the zone layout.

In our implementation zones are selected with the left stick and sub-zones with the right stick.

Selection keyboards sometimes use buttons on the game controller as shortcuts for space, backspace or shift. These shortcuts could improve TwoStick performance as well, at least if the thumbs do not need to leave the sticks to operate the buttons. We used the buttons under the forefingers for shortcuts. Left bumper button was used for backspace and the right bumper button on for space.

3.4 Optimal Expert Behavior

Optimal error free use of TwoStick consists of the following behaviors:

- Selecting zones and characters in parallel i.e. deflecting both sticks at the same time. Only re-centering needs to be carried out in the correct order with the zone stick being the last to move.
- The motion of the zone stick towards the next zone starts directly from the zone of the previous character without returning to the center.
- If two consecutive characters are in the same left stick zone only the character selection stick moves.

Hence, every character can be entered using two consecutive stick movements. During the first movement one or two sticks are moving. During the second movement only the character selection stick moves. All movements are simple straight lines. At the end of each movement the stick can either rest against the edge of its movement area or be released in the center position. Such movements should be very easy and fast

4 Experiment Design

After completing the TwoStick design, we needed to select the method of its evaluation. In general we have two ways of evaluating text entry methods: experiments and modeling. If an estimate on expert performance is needed, experiments are expensive. Modeling, on the other hand is fairly inexpensive if a valid model is available.

Looking at a mini-stick on a game controller we find edges and springs influencing the interaction. Consequently there are doubts regarding the validity of Fitts’ law and its derivatives (Steering, Crossing, etc.) as tools for this task. Additionally the stick movements needed for modeling TwoStick and its competitors vary a lot. For example Quikwriting requires continuous gesturing with round and pointed sweeps whereas the selection keyboards

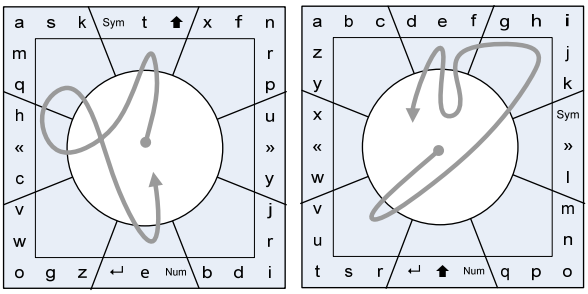


Figure 4. Quikwriting layout that was used by Isokoski and Raisamo (left) and XNav layout (right). The grey arrow represents the stroke needed for entering the word 'the'.

require straight line stick movements and button presses. To be reliable, a model would need to be calibrated with realistic usage data for each kind of stick movement. Given our limited resources we opted to skip modeling and go directly into collecting data on TwoStick text entry performance.

However, we still needed to select a representative technique to compete with TwoStick in the experiment. In this section we discuss the techniques that we considered.

4.1 Selection Keyboards

A *selection keyboard* is the de facto standard in gamepad text entry. Therefore, a selection keyboard is an ideal candidate for a comparison with TwoStick. From our perspective it is important to have a baseline to compare with. Because no longitudinal data for selection keyboard usage has been published, our experiment would serve as a baseline measurement for this technique as well.

When selection keyboards are operated with a game controller the interaction is split into navigating to a character and entering it by pressing a button. To better understand selection keyboard interaction, we implemented a generic selection keyboard module in our text entry software package. In our implementation we used the left stick for navigation and the A button (see Figure 1) on the right for selecting a character.

Discrete jumping from key to key (i.e. cursor cannot be in between keys) seems to be the dominant method for cursor movement in commercial products. Regarding the number of possible movement directions our informal pilot studies showed that using 8-way navigation does not seem to work out even after a long training (more than 10 hours). Alignment errors on the long horizontal transitions make the cursor jump to a wrong line too easily. This tends to cause delays that are longer than the savings gained by the ability to move diagonally. Unfortunately, others [20, 23] do not specify the way of navigation used in their experiments. We assume that they used 4-way navigation. Commercial products, like the Xbox 360 selection keyboard, use 4-way navigation.

According to our observations the navigation to a character in selection keyboards happens as follows. If the character is in the same row or column as the previous character, the stick is deflected in the direction of the intended character. The selection cursor keeps moving at a fixed rate until the stick is re-centered when the target is met. When the character is not in the same line or row the user has to make a turn. Ideally users do not re-center the stick when turning, but move it directly to the next position.

The rate at which the cursor moves when the stick is fully deflected is crucial to the maximal attainable text entry rate. If the cursor moves slowly, high rates cannot be achieved. If the cursor moves too fast, it is difficult to stop on the desired key. In our software we used a movement rate of 150 ms per key.

Some selection keyboards allow wrapping, i.e. continuing the movement over the edge of the layout so that the cursor upon exiting on one side of the keyboard enters from the opposite side. This provides a shortcut to the other side of the keyboard. We supported wrapping in our implementation.

The number of steps between two characters is determined by the layout. We used the QWERTY layout (see Figure 5). The QWERTY layout is well known by most game console users.

A different kind of a shortcut can be implemented by mapping some characters or functions to buttons on the gamepad. We designed space (right bumper), backspace (left bumper) and shift (Y) as button shortcuts.

4.2 Split Selection Keyboards

Split selection keyboards are interesting for comparing against TwoStick, because for novice users they are faster [20] than traditional selection keyboards. Wilson and Agrawala [20] report that with some practice they could write at rates exceeding 13 wpm.

Our split selection keyboard implementation followed the design reported by Wilson and Argwala [20]. Our experience was consistent with earlier reports in that the split keyboard is faster for novice mainly because fewer steps are needed to reach the characters. There are cognitive costs due to the need to switch attention between the two halves of the keyboard as well as potential gains due to parallel operation of the sticks. Novice users tended to enter characters sequentially and not make intentional parallel movements of the sticks. After some training the split selection keyboard was still faster, but we did not experience a big difference compared to normal selection keyboard text entry speed. Writing was cognitively more demanding, because of having two foci of attention. These costs reduced the positive effects of the split design (parallel movements, fewer steps).

4.3 Multi-Tap Selection Keyboard

Multi-tap selection keyboards mimic the well known multi-tap text entry method used in mobile phones. Each button on the selection keyboard is associated with more than one character. A character is entered by pressing the associated key once for the first character on the key, twice for the second, etc. If the next character is on the same key, a segmentation technique is needed. On mobile phones users usually have both time-out segmentation (1.5 seconds [16]) and a segmentation key. Multi-tap selection keyboards operate the same way except that a pointer on a virtual keyboard is used instead of a finger on a physical keyboard.

While scientific publications on multi-tap selection keyboards are missing, it appears that they are destined to be very slow text entry techniques. Our preliminary experiments with our own implementation left us mystified as to why console manufacturers consider multi-tap selection keyboards worth implementing. One possible explanation is their familiarity to mobile phone users, and the consequent initial positive response from these users. As mentioned in the introduction, some console manufacturers are planning to add a dictionary based disambiguation and possibly word prediction to their systems. These improvements might make multi-tap selection keyboards at least competitive with other methods. TwoStick could, of course, be enhanced with word prediction as well.

4.4 Quikwriting

We also wanted to compare TwoStick to *Quikwriting*. Quikwriting is the only method for which prior longitudinal data in gamepad use was available.

Examples of Quikwriting input are shown in Figure 4. When entering the letter 't' with the XNav layout the user first selects zone 7. The character is entered when the user returns to the center zone (zone 5). To enter 'h' the user first enters zone 3 and then continues to zone 2, because 'h' is in the upper field of zone

3. When the user moves the stick from zone 2 to zone 5 it will result in 'h' being entered.

Every token on the layout can be entered by either moving the stick in one zone and back or by entering a zone and navigating through other zones and returning the stick to the center.

Quikwriting can be used with two sticks. Isokoski and Raisamo [6] had a different layout with numbers and punctuation for the second stick. However, they found that users did not use the second stick much. Another way of utilizing the second stick in Quikwriting is to use both sticks on the primary alphabetic layout. The sticks can move in parallel as long as returning to the center happens in the correct order of the characters to enter.

One of the authors spent several weeks (1-2 hours per day) training with an implementation of two-stick Quikwriting. He did reach higher speeds than with one-stick Quikwriting, but the amount of training needed was a serious deterrent to recommending two-stick Quikwriting as a general solution for gamepad text entry.

Overall we concluded that single stick Quikwriting would be an interesting competitor for TwoStick.

4.5 EdgeWrite

Joystick operated *EdgeWrite* outperforms a selection keyboard in novice use and satisfaction [23]. EdgeWrite use has been studied extensively. Therefore, although EdgeWrite is a serious contender for TwoStick, doing yet another experiment with EdgeWrite is not as interesting as for example experimenting with two-stick Quickwriting would have been.

4.6 Dasher

Dasher [19] is a system where the user navigates through a tree of all possible texts using a pointer. The Dasher display zooms infinitely as the user steers the direction of the zoom. As characters are passed on the way, they are saved, and thus text is formed. Dasher is probably relatively fast in joystick use. Mouse users have reached speeds exceeding 30 wpm. One could expect that with a joystick at least 20 wpm is possible after sufficient training.

We implemented a game controller mode for using the Dasher implementation available on the web. Using Dasher with a joystick is definitely possible. We did not train enough to know the limits of our performance in gamepad operated Dasher writing. Overall, Dasher was found to be a serious contender for the fastest game controller compatible text entry method.

4.7 Selecting the Yard Stick

As described above, we had a number of text entry methods that we should compare TwoStick with. Obviously one experiment would not be sufficient for exhaustive ranking of all methods. We wanted to conduct a longitudinal experiment and considered two text entry methods to be the theoretical lower limit for how many text entry methods should be included. Unfortunately, we also considered it to be the practical upper limit. Longitudinal experiment calls for a within-subjects design to achieve adequate statistical power without needing to include too many participants. We believe that learning three or more new text entry methods would be too much to ask from our volunteer participants.

After a considerable debate, we ended up comparing a traditional QWERTY selection keyboard with TwoStick. Selection keyboards are the most widely used text entry method for game controller text entry. This is a sufficient justification for including selection keyboards in the experiment. The unfortunate consequence is that several interesting comparisons were excluded. However, the selection keyboard can work as a useful yard-stick. The other methods can be compared according to their performance relative to the selection keyboard. A method that is 50% faster than a selection keyboard is likely to be faster than another method that is only 20% faster. Over time the number of methods to compare to will explode. Conducting all comparisons is not economically possible in the long run.

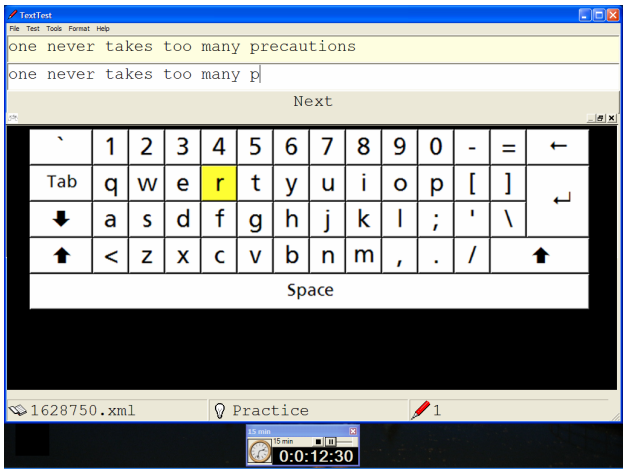


Figure 5. The display during the experiment. (The visualization of the keyboard has been enlarged for a better presentation)

5 Evaluation of TwoStick

We decided to compare TwoStick to the QWERTY selection keyboard. Our expectation was that QWERTY would be faster in the beginning but that TwoStick would be quickly learned and would, therefore, soon outperform the QWERTY selection keyboard.

5.1 Participants

For the experiment we recruited 8 volunteer participants (4 female, 4 male, all right-handed) between 22 and 29 (mean = 24.4, SD = 2.3) years of age. None of the participants had previous experience with TwoStick, but all used a desktop QWERTY keyboard in their daily work. Five participants had prior game controller experience (3 novices, 1 intermediate and 1 expert). Only one owned a game console, the others had only tried a few times or played as a kid. The participants had good English reading and writing skills, but they were not native English speakers.

5.2 Apparatus

We conducted the tests in a laboratory using a standard desktop computer. A Microsoft Xbox 360 controller for Windows served as input device. We implemented TwoStick and the QWERTY selection keyboard in C# using DirectInput. Both methods had shortcut keys for backspace (left bumper), space (right bumper), and shift (Y button). The resolution of the display was set to 1280x1024. The size of the smallest buttons was 40x40 pixels. We used the Frutiger Roman font in 20 pt size for the key labels.

A separate application (TextTest 2.1.4 [21]) was used to present the phrases to transcribe and save input stream for later analysis. Furthermore, every text entry method generated a log file with all button presses and stick movements. The arrangement of the windows on the screen is shown in Figure 5. The results were analyzed using a character-level error analyzer by Wobbrock et al. (StreamAnalyzer 2.0.2 [21]) and custom made scripts.

5.3 Task

The task consisted of transcribing phrases presented on the display. Entering a phrase was finished by entering the "enter" character with the text entry method in use. Participants were

instructed to enter the text as fast as possible while not making too many errors. They could correct errors using backspace, but were allowed to leave errors in the transcription.

The phrases were selected in a random order from a phrase list based on the 500 phrase list by Soukoreff and MacKenzie [10]. We modified the phrases by adding upper case characters and punctuation where it seemed grammatically appropriate. The reason of this modification was the increasing number of publications arguing that transcribing text consisting only of lower case alphabet is not representative for text entry in general [6, 25].

The corpus was in English but our participants were native Finnish, German, Tamil and Telugu speakers. Isokoski and Linden [4] showed that participants are faster when transcribing text in their native language. However, in the interest of maintaining compatibility with the majority of text entry work, we decided to use the English language.

5.4 Procedure

Each participant completed 20 sessions of text transcription. Each session consisted of two 15-minute sub-sessions. One sub-session was for TwoStick and the other for selection keyboard text entry. We counterbalanced the order of the text entry methods to neutralize the effects of learning and fatigue. For each session half of the participants began with one method and the other half with the other. The TextTest application showed how many phrases had been entered. Thus, the participants were able to observe the development of their text entry speed at the end of each session.

The experiment started with an explanation of its purpose. After this, a questionnaire for participant information and text entry background was completed. Before the first use of each text entry method, its use was explained orally to the participant. Participants' questions were answered, but they were not allowed to use the systems before the first session began. After the first session the participants were interviewed and they filled out a questionnaire about their impressions on the text entry methods.

After the 15-minute TwoStick block in the last session the visualization of the TwoStick layout was turned off and participants were instructed to transcribe text for a further 10-minutes. The participants could only see the presented and transcribed text and the status display showing the shift mode. Our purpose was to see if eyes-free use of TwoStick is possible. The same questionnaire that was used at the end of the first session was repeated at the end of the last session.

The experiment was designed to have the text entry method (two levels) and the amount of training (20 levels) as the independent variables. The dependent variables were text entry rate and error rate (total, corrected, and uncorrected). The main effects were tested using a 2x20 (method, session) repeated measures ANOVA. Further pair-wise comparisons were done using paired samples t-tests. Questionnaire ratings were compared using the non-parametric Wilcoxon's signed ranks test.

6 Results

6.1 About the Data

The participants transcribed in total 8,853 phrases over 74.87 hours, which resulted in 297,485 characters in the input stream for further analyses.

Sometimes during the transcription participants accidentally pressed the enter button before they had entered the whole phrase. This behavior resulted in a high error rate for that phrase. These incidents were nearly equally divided between the two methods (4 TwoStick, 5 QWERTY phrases, 0.1% of the total number). Because they were anomalies due to the experimental setup and not representative of normal text entry behavior we decided to remove these phrases from further analysis.

Due to failed coordination between the sites, we forgot to present the questionnaire at the end of the first session to one participant. Consequently these results for that participant are missing from the following report.

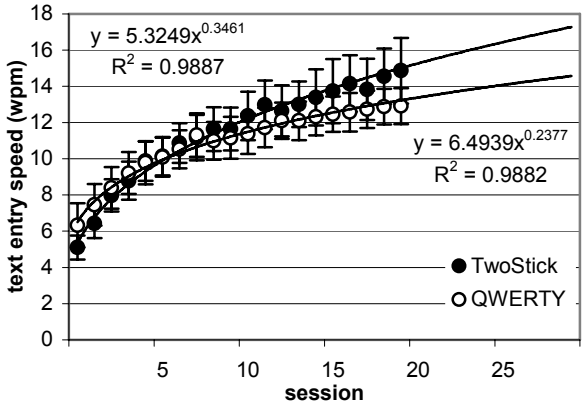


Figure 6. Average text entry speed for TwoStick and QWERTY selection keyboard. The error bars show standard deviations.

6.2 Speed

The development of text entry speed is shown in Figure 6. There was no overall effect of text entry method on text entry speed. However, the effect of session ($F_{19,133} = 88.67$, $p < .001$) was statistically significant meaning that performance improved over time. The interaction of method and session was also statistically significant ($F_{19,133} = 5.38$, $p < .001$) meaning that the learning progressed at different rates for the two methods.

In the first session the participants were statistically significantly (t (7)= 2.51, $p < .05$) faster with QWERTY (6.32 wpm, SD = 2.44) than with TwoStick (5.10 wpm, SD = 1.33). As Figure 6 indicates there was a cross-over of entry speed during the experiment. During the last session TwoStick (14.87 wpm, SD = 3.62) was statistically significantly faster (t (7) = 2.83, $p < .05$) than QWERTY (12.90 wpm, SD = 2.00). The fastest participant achieved a mean text entry rate of 21.24 wpm during the last session. The fastest session mean with the QWERTY selection keyboard was 15.13 wpm. It was achieved in session 18 by a different participant.

6.3 Error Rates

We used the error analysis metrics by Soukoreff and MacKenzie [17] and the analysis software by Wobbrock and Myers [21]. It classifies errors in two groups: 1) the *uncorrected errors* are errors that were left in the transcribed phrases and 2) *corrected errors* that are errors the participants made, but corrected while transcribing. *Total error rate* is the sum of these two.

6.3.1 Total Error Rate

We found a significant effect of text entry method on total error rate ($F_{1,7} = 7.17$, $p < .05$) meaning that TwoStick overall was more error-prone than QWERTY. The effect of session ($F_{19,133} = 3.34$, $p < .001$) was statistically significant, showing that over time the total error rate decreased. The interaction of session and method was not statistically significant.

In the first session participants made significantly (t (7) = 3.88, $p < .01$) more errors with TwoStick (13.34%, SD = 4.93) than with QWERTY (8.58%, SD = 4.41). In the last session there was no statistically significant difference in total error rate between TwoStick (8.21%, SD = 4.43) and QWERTY (5.35%, SD = 1.91).

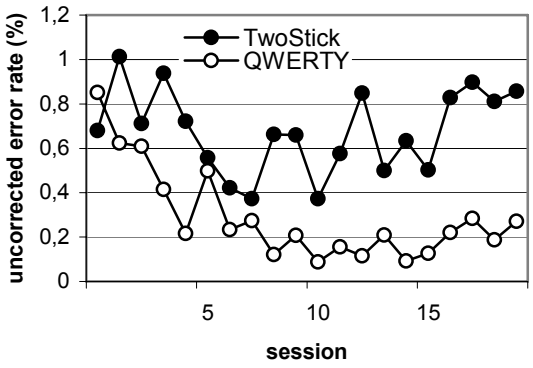


Figure 8. Average uncorrected error rate for TwoStick and QWERTY selection keyboard as a function of session number.

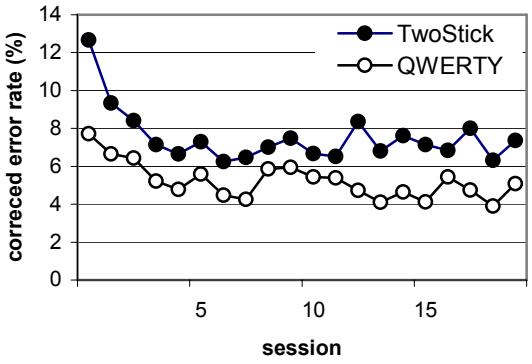


Figure 7. Average corrected error rate for TwoStick and QWERTY selection keyboard as a function of session number.

Over all sessions the total error rate was 8.20% (SD = 1.52) for TwoStick and 5.52% (SD = 1.13) for QWERTY.

6.3.2 Corrected Error Rate

The effect of text entry method on corrected error rate was statistically significant ($F_{1,7}$ = 5.62, p < .05), meaning that participants correct a different number of errors with different methods. The effect of session for corrected error rate was also statistically significant ($F_{19,133}$ = 3.26, p < .001), showing that over time the corrected error rate decreased (see Figure 7). The interaction of session and method was not statistically significant.

In the first session corrected error rate was statistically significantly higher (t (7) = 4.31, p < .01) with TwoStick (12.67%, SD = 4.47) than with QWERTY (7.73%, SD = 3.19). In the last session there was no statistically significant difference between TwoStick (7.36%, SD = 4.42) and QWERTY (5.08%, SD = 1.83).

During the first session participants corrected 94.90% of all errors with TwoStick and 90.04% with QWERTY. In the last session 89.58% of all errors made with TwoStick and 94.91% of QWERTY errors were corrected.

Over all sessions the corrected error rate was 7.52% (SD = 1.45) for TwoStick and 5.23% (SD = 0.97) for QWERTY.

6.3.3 Uncorrected Error Rate

The effect of method on uncorrected errors (see Figure 8) was statistically significant ($F_{1,7}$ = 8.24, p < .05). The participants left more errors in the text when using TwoStick. The effect of session and the interaction of method and session were not statistically significant.

Further t-tests between the methods during the first and the last sessions showed no statistically significant differences. In the first session uncorrected error rate of TwoStick was 0.68% (SD = 0.68) and with QWERTY 0.85% (SD = 1.74). In the last session uncorrected error rate for TwoStick was 0.86% (SD = 1.15) and for QWERTY 0.27% (SD = 0.47).

Over all sessions the uncorrected error rate was 0.67% (SD = 0.19) for TwoStick and 0.29% (SD = 0.21) for QWERTY.

6.4 Eyes Free Entry with TwoStick

The 10-minute trial in TwoStick use without the visualization resulted in an average text entry rate of 7.9 wpm (SD = 3.5) with a total error rate of 24.23% (SD = 13.3). Participants corrected 98.2% of all errors which led to a corrected error rate of 23.92% (SD = 13.26) and an uncorrected error rate of 0.32% (SD = 0.97). These numbers show that eyes-free text entry with TwoStick is possible even without explicit eyes-free training.

6.5 Questionnaires and Interview Data

The results of the questionnaires after the first and last session can be seen in Table 2. The values in the two bottom rows of Table 2 show that participants preferred QWERTY in the beginning for short and long text entry. After the trainings sessions they preferred TwoStick.

Within the first session there were no statistically significant differences between the methods. In the last session we found a significant difference within participants stating that they were faster (z = 2.46 p < .05) and had more fun (z = 2.41 p < .05) using TwoStick. The numbers for the questions with statistically significant differences are shown in bold in Table 2.

The findings above agree with the interview data. In the beginning QWERTY was preferred, but most of the participants recognized the potential of TwoStick being faster. With training the QWERTY selection keyboard became boring. Participants also felt that they soon reached the limits of their ability to advance in QWERTY use.

7 Discussion

7.1 Learning and Errors with TwoStick

The main reason for getting faster was that users learned the layout and needed less and less visual search to complete the task. Over time stick movement became continuous. It might be possible to support the visual search and learning of the layout by separating the zones more clearly by using a wider spacing between the zones.

Wrong timing of stick movements was the greatest cause of errors. Looking at the confusion matrices produced by the analysis software by Wobbrock and Myers [21], we found that participants started moving the left stick before centering the right stick.

	Session 1		Session 20	
Scale 1-5	**TwoStick**	**QWERTY**	**TwoStick**	**QWERTY**
Difficult-Easy	3.4 (1.1)	3.9 (0.9)	3.9 (1.0)	3.9 (1.1)
Pain-Enjoyment	3.1 (0.9)	3.1 (0.9)	3.5 (0.9)	2.8 (0.5)
Slow-Fast	2.3 (1.3)	3.1 (1.3)	**4.2 (1.0)**	**3.1 (0.8)**
Many-No Errors	2.1 (1.1)	3.0 (1.2)	1.8 (1.0)	2.6 (0.7)
Bored-Fun	3.4 (0.5)	3.4 (0.8)	**4.1 (0.6)**	**2.8 (1.0)**
T-Q short phrases	3.3 (1.6)		2.4 (1.4)	
T-Q long phrases	3.4 (1.5)		1.8 (1.1)	

Table 2. Means (and standard deviations) of the questionnaires. The scale is from 1 associated with the left extreme to 5 associated with the right extreme. Last two rows show the ratings for different phrase lengths.

Consequently, they entered characters in the wrong zone, but right sub-zone (e.g. 'n' instead of 'e').

Another common error type was missing a zone by one. For example, an upwards move ending up in one of the upper corner zones instead of the middle. Wobbrock et al. [23] report that thumb dexterity and range of motion along one diagonal is better than along the other diagonal. It might be possible to reduce error rates by rotating the input area or adjusting the zone borders.

7.2 Comparing TwoStick Performance to Other Methods

Isokoski and Raisamo [6] reported a text entry rate of 13 wpm with Quikwriting. Our participants achieved 14.9 wpm with TwoStick. The experiments were not exactly the same. In the Quikwriting study the participants used Quikwriting with two input devices learning the Quikwriting layout all the time. Our participants used different layouts, but had training in the use of mini-sticks all the time. Novice performance with Quikwriting was 4 wpm. With TwoStick we measured 5.1 wpm. We suspect that TwoStick might be easier to learn than Quikwriting.

Comparing our measured novice selection keyboard performance (6.32 wpm) to Wilson and Agrawala [20] (6.48 wpm) shows only a minute difference. Split selection keyboards may be faster for novice users (Wilson and Agrawala report 7.08 wpm for QWERTY), but longitudinal data is needed to measure whether the split design shows a good learning curve. The anecdotal evidence from Wilson and Agrawala and Wobbrock et al. as well as our results and the Quikwriting results of Isokoski and Raisamo all show text entry rates in the range of 13-15 wpm for joystick-based gamepad text entry. More work is needed to find out if some of these methods are better than the others.

7.3 TwoStick and Other Input Devices

TwoStick can easily be adapted for use with any input device capable of producing nine different inputs. As mentioned earlier, there are similarities to Quikwriting (originally a stylus-based system) and to TNT [3] (originally a keypad based system). With TNT users could write at 9.3 wpm (novice) and at 17.7 wpm (after 7.5 hours of training)[1]. Game consoles are often shipped with a gamepad and a remote control. A TwoStick/TNT system can be used for text input with both input devices.

8 Conclusion And Future Work

We presented TwoStick, a novel text entry method for game controllers. In an experiment we showed TwoStick to be faster in expert use than a QWERTY selection keyboard.

Independently from our research a hardware add-on called Texter [1] was recently announced. Texter is similar to TwoStick, but works without an on-screen visualization. We greet this product with joy since it makes our results timely and potentially valuable for those who consider purchasing this product.

Now that we have established that TwoStick can outperform the de-facto standard in gamepad text entry, we plan to continue the search for the most efficient gamepad text entry method by comparing in more detail those methods that have achieved or are likely to achieve the same. This group includes Edgewrite, Dasher, Quikwriting, Split QWERTY and possibly others.

ACKNOWLEDGMENTS

We thank Peter König and Marko Sirka for help with the implementation and the participants for their time and enthusiasm in the experiment.

[1] Note that Ingmarsson et al. report the peak result for the best one-minute window in each session. The mean text entry rate over the whole session is lower.

Referernces

[1] Blue Orb Inc., Texter. http://www.blueorb.com/gaming/whatis.cfm

[2] Co-operwrite. myText. (1997).http://www.my-text.com/

[3] Ingmarsson, M., Dinka, D. and Zhai, S. TNT: a numeric keypad based text input method. *In Proc. CHI 2004*, ACM Press (2004), 639-646.

[4] Isokoski, P. and Linden, T. Effect of Foreing Language on Text Transcription Performance: Finns Writing English. *In Proc. NordiCHI 2004*. ACM Press (2004), 105-108.

[5] Isokoski, P. and Raisamo, R. Device independent text input: A rationale and an example. *In Proc. AVI 2000*, ACM Press (2000), 76-83.

[6] Isokoski, P. and Raisamo, R. Quikwriting as a multi-device text entry method. *In Proc. NordiCHI 2004*, ACM Press (2004), 105-108.

[7] Kanellos, M. Microsoft scientists pushing keyboard into the past. CNET News.com (2006), May 3, 2006.

[8] KeyStick. http://www.n-e-ware.com/KeyStick.htm

[9] Koleman, M. Weegie Homepage. (2001). http://weegie.sourceforge.net/

[10] MacKenzie, I. S. and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. *In Ext. Abstracts CHI 2003*, ACM Press (2003), 754-755.

[11] MacKenzie, S. Mobile text entry using three keys. *In Proc. of NordiCHI 2002*, ACM Press (2002) 27-34.

[12] MobileQWERTY. (2006). http://www.mobience.com/

[13] Perlin, K. Quikwiriting for the Xbox. http://mrl.nyu.edu/~perlin/experiments/xq/

[14] Perlin, K. Quikwriting: continuous stylus-based text entry. *In Proc. UIST 1998*. ACM Press (1998), 215-216.

[15] Proschowsky, M., Schultz, N. and Jacobsen, N.E. An intuitive text input method for touch wheels. *In Proc. CHI 2006*. ACM Press (2006), 467-470.

[16] Silfverberg, M., MacKenzie, I.S. and Korhonen, P. Predicting text entry speed on mobile phones. *In Proc. CHI 2000*, ACM Press (2000), 9-16.

[17] Soukoreff, W. R. and MacKenzie, S. I. Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. *In Proc. of CHI 2003*. ACM Press (2003), 113-120.

[18] Venolia, D. and Neiberg, F. T-Cube: a fast, self-disclosing pen-based alphabet. *In Proc. of CHI 1994*. ACM Press (1994), 265-270.

[19] Ward, D.J., Blackwell, A.F. and MacKay, D.J. Dasher - data entry interface using continuous gestures and language models. *In Proc. of UIST 2000*, ACM Press (2000). 129-137.

[20] Wilson, A. D. and Agrawala, M. Text entry using a dual joystick game controller. *In Proc. CHI 2006*. ACM Press (2006), 475-478.

[21] Wobbrock, J.O. and Myers, B.A. Analyzing the input stream for character-level errors in unconstrained text entry evaluations. *ACM TOCHI*, 13, 4 (2006), 458-489.

[22] Wobbrock, J.O. and Myers, B.A. Gestural text entry on multiple devices. *In Proc. ASSETS 2005*. ACM Press (2005), 184-185.

[23] Wobbrock, J.O., Myers, B.A. and Aung, H.H. Writing with a joystick: A comparison of date stamp, selection keyboard and EdgeWrite. *In Proc. GI 2004,* CHCCS (2004), 1-8.

[24] Zeller, T. A Great Idea That's All in the Wrist. (June 5, 2005). http://www.nytimes.com/2005/06/05/ weekinreview/05zeller.html?hp

[25] Zhai, S., Hunter, M. and Smith, B. A. Performance Optimization of Virtual Keyboards. *Human-Computer Interaction*, 17, Lawrence Erlbaum (2002), 229-269.

[26] Zi Corporation, eZiText word prediction. http://www.zicorp.com/eZiText.htm