Writing with a Joystick: A Comparison of Date Stamp, Selection Keyboard, and EdgeWrite

Jacob O. Wobbrock, Brad A. Myers and Htet Htet Aung
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
{jrock, bam, hha}@cs.cmu.edu
http://www.cs.cmu.edu/~edgewrite/

Abstract

A joystick text entry method for game controllers and mobile phones would be valuable, since these devices often have joysticks but no conventional keyboards. But prevalent joystick text entry methods are slow because they are selection-based. EdgeWrite, a new joystick text entry method, is not based on selection but on gestures from a unistroke alphabet. Our experiment shows that this new method is faster, leaves fewer errors, and is more satisfying than date stamp and selection keyboard (two prevalent selection-based methods) for novices after minimal practice. For more practiced users, our results show that EdgeWrite is at least 1.5 times faster than selection keyboard, and 2.4 times faster than date stamp.

Keywords: Text entry, text input, joystick, game controller, game console, physical edges, corners, gestures, unistrokes.

1 Introduction – Why Joystick Text Entry?

Joysticks have served as input devices since the earliest computers [7]. The two-player version of *Computer Space*, the first coin-operated arcade game, used two mounted joysticks in 1972. In 1978, Atari released its first game console, the Atari 2600, which had no keyboard, just a joystick. Joysticks have been studied in human-computer interaction since at least the seminal study by Card et al. in 1978 [1]. Yet despite joysticks' considerable tenure, no satisfying text entry techniques have been developed for them. The methods that do exist are mostly selection-based; they require screen real-estate to display options, are difficult to use without looking, are hard to customize, and are slow, requiring many movements per character.

Today's computer game industry would benefit from better text entry for game consoles, which often have only game controllers as input devices; if they have keyboards at all, they are sold separately at extra cost. Many game consoles are now networked, and require extensive text entry during configuration before they allow game play. For example, registration for the Xbox *Live!* service requires entering personal and billing information and can take more than 30 minutes using a joystick and an on-screen selection keyboard. Furthermore, many networked games allow for communication among players using short bursts of instant messenger-style text. With only selection-based text entry methods for game controllers, this can be awkward.

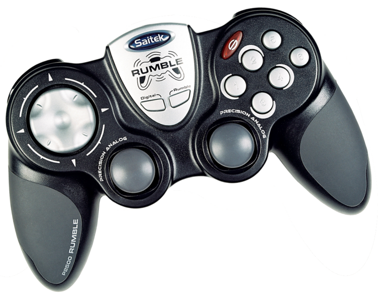


Figure 1. The Saitek P2500 Rumble Force Pad. Our experiment used the two thumbsticks and one of the silver buttons.

Mobile devices have also placed high demands on text entry development. Numerous text entry methods have been investigated, including those driven by buttons, character recognition, virtual keyboards, thumbwheels, and voice. Many new handheld devices, such as the Ericsson T68i mobile phone, are equipped with miniature joysticks for navigation and selection purposes, yet have no capability for joystick text entry. Joystick text entry on mobile devices reduces the need for screen areas devoted to stylus entry, for virtual keyboards that take up precious screen real-estate, and for multiple button-taps to select desired characters. They also can be used without looking, which may have positive implications for blind use.

Another potential use of joystick text entry is for users of power wheelchairs. Technology is already commercially available [22] to enable a person to control a computer's mouse from a power wheelchair joystick, but options for text entry are limited to mouse-based selection techniques, like the WiVik on-screen keyboard [19].

In this paper, we present a new joystick text entry method that is not based on selection, but on gestures. We use the EdgeWrite alphabet [27], originally a stylus-based text entry method for users with motor impairments. The properties of this alphabet make it well-suited for text entry with joysticks. Our experiment shows that joystick EdgeWrite is faster, produces more accurate phrases, and is more satisfying to users than date stamp or selection keyboard, two prevalent selection-based methods.

2 Challenges for Writing with a Joystick

Joysticks, like those found on game controllers (Figure 1), commonly operate in one of two ways: position-controlled or rate-controlled. With position-control, the physical range of the joystick is mapped to a plane (e.g., the screen), and the position of the stick corresponds to a position in the plane. Joystick-driven screen magnifiers have been designed using position-control [11]. With rate-control, on the other hand, the further the stick is moved from its center, the faster the position changes. Rate-control is common in first-person games and for joystick-controlled mouse cursors [13].

It would seem that position-control might be the ideal candidate for "writing" with a joystick, as a user could trace an (x, y) path like she does with a stylus on a PDA. Many studies, however, confirm that joysticks are not as accurate for positioning as mice, trackballs, touchpads, and tablets (e.g., [1, 5, 15, 18]). Indeed, our design explorations confirm the difficulty of making smooth letter-forms using a joystick. The prospect of writing in an alphabet like Graffiti is therefore dubious. If gestures are to be used, they will have to be designed to overcome this difficulty.

Human physiology also complicates joystick text entry. For example, the dexterity of the thumb changes with its position relative to the hand, causing changes in range of motion [8]. The index finger has the highest Fitts' index of performance [12], making it better suited than the thumb for control tasks [4]. The velocity of a writer depends on whether she moves her arm or only her wrist, and upward strokes are generally faster than downward ones [9]. Some results show that humans have a difficult time returning a joystick to the same position it was before [11]. Other results show humans often under- or overshoot their targets while using joysticks, and that joystick movement can be tremulous, comprised of sub-movements and repeated in-path corrections [17]. Such variables may subvert any attempts at "writing" with joysticks.

While not a panacea, EdgeWrite is well-suited to overcoming many of these challenges. The next section explains why.

3 A New Method Based on EdgeWrite

The EdgeWrite text entry method was invented to help people with motor impairments enter text with a stylus on a PDA. Many people with motor impairments have difficulty writing Graffiti because of their inability to make smooth curves and straight lines due to tremor or rapid fatigue [25]. The stylus version of EdgeWrite addressed these problems by offering a more accurate and physically stable means of text entry through the use of *physical edges* [26]. In fact, all stylus entry in EdgeWrite is performed within the confines of a small plastic square, and all strokes are along the edges or diagonals and into the corners of this square. Recognition works not by analyzing the path of movement, but by examining the order in which the corners of the square are hit. When compared to Graffiti, EdgeWrite was hugely more accurate for some people with motor impairments, and at least 18% more accurate for able-bodied users (p<.02). It was also found to be just as learnable as Graffiti and about as fast. A detailed discussion of stylus EdgeWrite is available elsewhere [27].

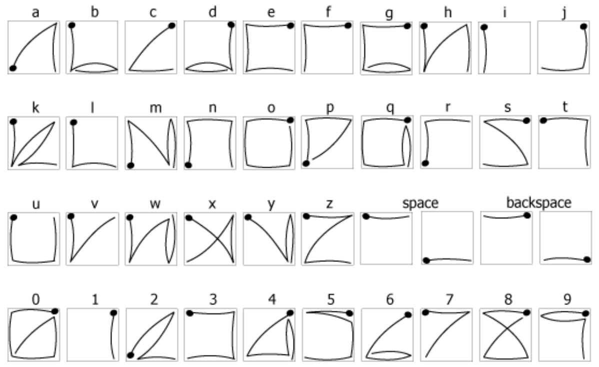


Figure 2. The EdgeWrite alphabet. Alternate forms exist for most characters (not shown). The bowing of line segments is only illustrative and does not depict actual movement, which is in straight lines. For more detail, see [27].

3.1 The EdgeWrite Alphabet

The EdgeWrite alphabet has properties that make it well-suited to meeting the challenges of joystick text entry. The alphabet is unique among unistroke methods in that every character is comprised of up to six well-defined segments between the vertices of a square (i.e., the four sides and the two diagonals). Thus, all motion in EdgeWrite is ideally in straight lines between corners. But straight lines are not required for gesture recognition, since recognition depends not on the path of movement but instead on the sequence of corners that are hit.

EdgeWrite can be easily implemented on any surface that has a square area bounded by physical edges. Physical edges provide a Fitts' Law benefit, as they allow for "target overshoot" without sacrificing accuracy [26]. Physical edges also provide tangible feedback during movement and result in greater speed and stability of motion [25].

Joysticks are usually best used for control, not positioning. But EdgeWrite's use of stabilizing physical edges allows joysticks bounded by square areas to be used in position-control mode for writing EdgeWrite characters. The areas bounding the thumbsticks on the Saitek P2500 (Figure 1) are squares with slightly rounded corners. In our study, we used this joystick without modification.

With a joystick it is difficult to make smooth characters, such as those required by Graffiti, but EdgeWrite characters are easy to make by pushing the stick from corner to corner within the plastic bounding area. Edges naturally guide the stick, and corners naturally pocket it, making accurate motions easy. As shown in Figure 2, EdgeWrite characters begin in one of four corners, easily accessed from the center of a square with a self-centering joystick.

Isokoski [9] offers a complexity measure designed to compare unistroke alphabets by abstracting their characters into composites of straight lines. EdgeWrite requires no such abstraction, as its characters are already comprised of straight lines. The complexity of the EdgeWrite letters in Figure 2 is 2.30, lower, and thus "faster," than Roman letters (2.76), Graffiti (2.54), and MDITIM (3.06), but higher than Unistrokes (1.40). EdgeWrite was shown to be as learnable as Graffiti, and is likely to be more learnable than Unistrokes [6] or MDITIM [10], since their letters do not generally resemble Roman letters, as EdgeWrite's do.

A final strength of EdgeWrite is that it is easy to customize. Only *one* "training example" is required from an end-user to teach EdgeWrite a new character since a sequence of corners is unambiguous. EdgeWrite is not a pattern matcher, so it does not need multiple prototypes for a training set. Selection-based methods, by contrast, are harder to customize since they require graphical options.

3.2 Design and Implementation

To understand how EdgeWrite works with a joystick, we must understand how EdgeWrite partitions the joystick's coordinate plane. Using C# and DirectInput, the joystick is to be polled for its position every 55 ms, which proved sufficiently often. The (*x*, *y*) position falls within the range of the *x*, *y* axes (-100, +100). In practice, none of the joysticks we used centered perfectly at (0, 0); some were off by as much as ±20.

EdgeWrite corners are triangular so that diagonal strokes do not accidentally hit them [27]. In pilot tests, using static corners as shown in Figure 3*a* proved to be inadequate because some subjects still accidentally hit unwanted corners when trying to make diagonals. For right-handed users, the problematic diagonal is from upper-left to lower-right (Figure 4). The other diagonal is not a problem. Left-handed users experience the opposite problem.

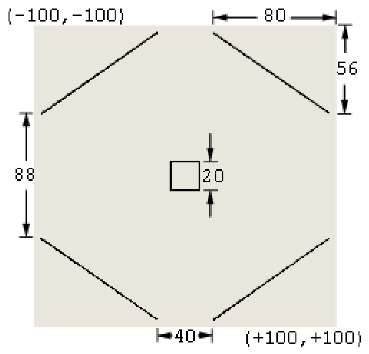


Figure 3a. Inflated dimensions of the joystick coordinate plane for a right-handed user. Corner areas are triangular so that accidental corner-hits when moving along a diagonal are rarer than they would be if the corners were rectangular.

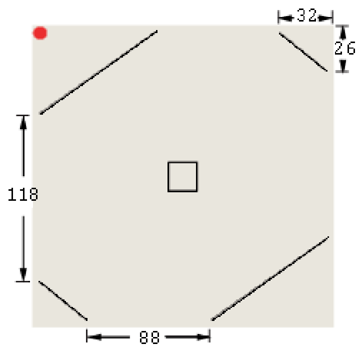


Figure 3b. Deflated dimensions of the joystick coordinate plane. The dot in the upper-left indicates the joystick position. Deflation gives more room for error on the hard diagonal stroke from upper-left to lower-right (for a right-handed user). See also Figure 4.

This difficulty arises because the thumb's dexterity and range of motion along one diagonal is much better than along the other diagonal. Figure 4 shows the thumb position of a right-handed user and the underlying joystick. The easy diagonal is along the natural arc of the thumb, while the difficult diagonal is along the length of the thumb itself.

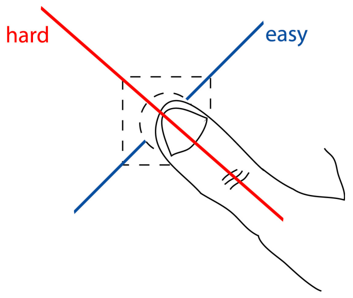
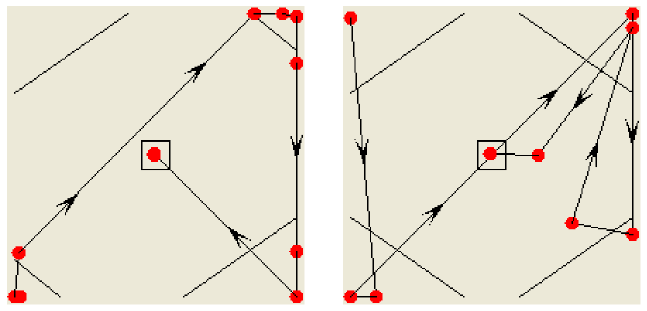


Figure 4. When on the joystick, a right-hand thumb is set so that one diagonal is easy while the other is more difficult. This is why we deflate two corners when the joystick is in danger of accidentally hitting them, as shown in Figure 3b.

We accommodate this difficulty by *deflating* the accidentally-hit corners when the joystick is in a corner subject to the problematic diagonal (Figure 3*b*). This allows users to be much sloppier without hitting an unwanted corner. If a user actually *wants* to hit a deflated corner, no harm is done, because sliding the joystick along the plastic edge of the bounding square is easy and accurate [25]. Deflated corners re-inflate once they are hit.

A design challenge is how to segment between letters. In unistroke text entry with a stylus, a pen-down event starts a character and a pen-up event ends it. There is no analog to this for a joystick. We built versions that used button presses and center dwell-time for segmentation, but both proved awkward. Instead, we segment characters by starting a character when a corner is entered, and ending it when the polling of the joystick yields two successive points in the center (Figure 5*a*). From a user's perspective, this means relaxing on the joystick so that it naturally snaps-to-center. With this scheme, annoying pauses are not necessary between characters, as they are with center dwell-time segmentation. In our testing, users did not notice any delays, and there were no observed segmentation errors.



Figures 5a, 5b. A clean trace of "a" (left) and a sloppy but recognized trace of "w" (right). The "w" is sloppy because it fails to snugly impact the bottom-right corner.

4 Prevalent Joystick Text Entry Methods

Here we describe the date stamp and selection keyboard methods of joystick text entry. We compare EdgeWrite to these two methods in our experiment in Section 6.

4.1 Date Stamp

The date stamp method is familiar to people who have entered their initials on the high-score screen of an arcade game. This method gets its name from a post office stamp that has rotating dials for each character.

While there are many variations on this method [14], ours uses the sequence (minus punctuation) from [23]. The sequence is [*space*][*a*..*z*][*0*..*9*](repeat). Moving the joystick down cycles the current character forward through the sequence (*a*→*z*). Moving the joystick up cycles the current character backward through the sequence (*z*→*a*). Moving the joystick right commits the current character and initializes a new stamp with "*a*." Moving the joystick left deletes the most recently committed character and initializes the stamp *with that character*. Thus, after deleting a letter, a user is not forced to start from "*a*" again. This makes under- and overshoots easy to correct.

If users hold the stick up or down, the date stamp cycles after an initial pause of 390.6 ms with a repeat delay of 62.5 ms. We took these values from keyboard key-repeat times.

4.2 Selection Keyboard

The selection keyboard method uses an on-screen keyboard over which a user moves a selection halo up, down, left, or right (Figure 6). When the user presses a joystick button, the currently-highlighted key is "pressed." When a key is pressed, the halo remains where it is and does not jump to a home position. The halo can wrap around the keyboard horizontally or vertically, staying in the same row or column. Key-repeat behavior, identical in timing to the date stamp method, governs rapid movement of the halo. Our layout is copied from selection keyboards from the Xbox *Live!* registration sequence and two popular Xbox games: *Halo* and *Brute Force*.

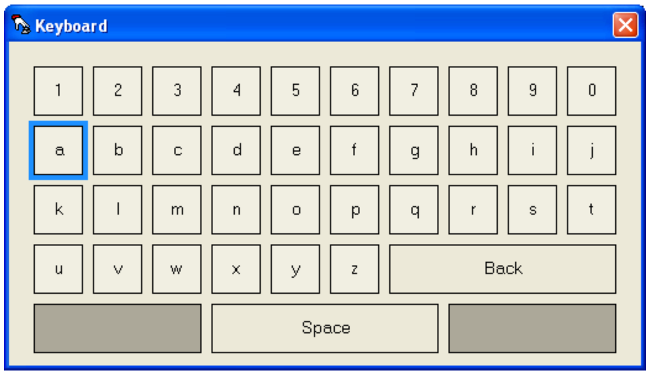


Figure 6. The selection keyboard used in our experiment. This keyboard was based on 3 selection keyboards from Microsoft's Xbox. Here the selector is positioned over the letter "a." The dark buttons are Xbox-specific and were not used in the study.

5 Related Work

The EdgeWrite technique is similar to other unistroke methods. These include the original Unistrokes [6], Graffiti, and MDITIM [10]. Like Graffiti, EdgeWrite has characters that are similar to Roman letters. EdgeWrite was made to avoid some problematic aspects of Graffiti; for example, by tolerating the presence or absence of initial down strokes on *b*, *d*, *m*, *n*, *p*, and *r*, or a final down stroke on *u*. EdgeWrite includes different forms of *k* to avoid the *k*-*x* confusion familiar to Graffiti users. It also avoids the necessity for two input regions, as all input occurs within a single square.

MDITIM [10] is "device independent" and designed to work on multiple platforms, including joysticks. EdgeWrite differs from MDITIM in that EdgeWrite characters may contain diagonals, but MDITIM characters use only *north*, *east*, *west*, and *south* primitives; EdgeWrite characters feel like Roman characters, but MDITIM's generally do not; and physical edges are integral to EdgeWrite, both in performance and recognition, but not to MDITIM.

Weegie [2] is a prototype joystick text entry method for use on X11. With Weegie, a user moves a joystick to various positions (e.g., 12 o'clock) to access different characters. EdgeWrite differs from Weegie in that EdgeWrite's strokes are similar to Roman characters, whereas Weegie's arrangement of letters has no mnemonic advantage, and EdgeWrite uses only one joystick, whereas Weegie uses two.

KeyStick [24] is a joystick text entry method for use on some mobile phones. With KeyStick, a user moves the joystick left, right, up, or down to access menus of characters. Like Weegie, the placement of KeyStick's characters is not reminiscent of Roman forms.

myText [3] is another method for joystick text entry on mobile phones. Unlike Weegie and KeyStick, myText is not position- or menu-based, but gesture-based like EdgeWrite. myText does not recognize characters by corner hit-testing like EdgeWrite, but by "unit vectors" of motion. No test results for unconstrained text entry are currently available.

Only recently have the algorithmic tools necessary for the analysis of unconstrained text entry experiments become available [21]. These tools allow us to compare text entry methods with different keystrokes per character [14]. The interested reader is directed to [21] for details.

6 Experimental Validation

6.1 Subjects

We recruited 18 subjects from the nearby university communities. The median age was 21.5. Four were female and 1 was left-handed. Thirteen indicated they had technical majors or occupations. Six had never used joysticks to play videogames and only 2 used joysticks daily. Only 1 was a daily PDA user. Ten had never tried Graffiti. Subjects were paid $20 US for a 90-minute test in which they entered text using 3 entry methods. No subjects had any prior experience with EdgeWrite.

6.2 Apparatus

We conducted tests in a laboratory using an 866MHz Pentium 3 machine running Windows XP with 256MB RAM. We used a 16"×12.4" Hitachi monitor set to 1280×1024 resolution and 32-bit color. We implemented the test software (Figure 7) in C# using DirectInput 9.0b. Our font was Microsoft Sans Serif 24-point, and our joystick was a Saitek P2500 Rumble Force Pad (Figure 1).

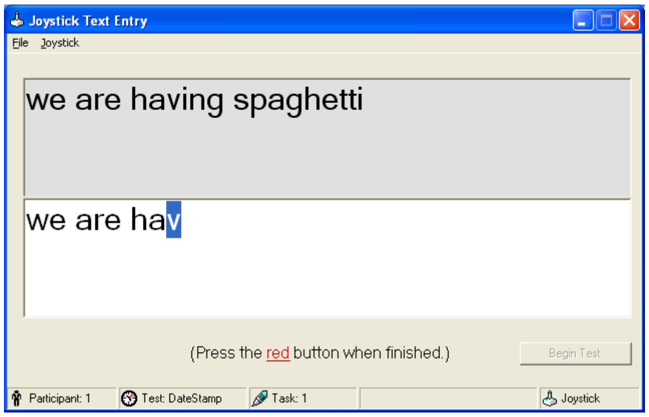


Figure 7. The text entry suite. The target phrase is shown at the top and the user's input is shown below it, here using date stamp.

6.3 Procedure

Subjects used EdgeWrite, date stamp, and selection keyboard in a single-factor within-subjects design. The entry methods were assigned to subjects in a fully counterbalanced order to neutralize learning effects and fatigue. Analyses of variance for test order show no significant differences.

Subjects practiced each method immediately before testing with it. Practice was designed to provide the minimum amount of proficiency needed to perform the technique. For date stamp and selection keyboard, this was just a single phrase (about 30 letters), as subjects found these methods trivial to learn. For EdgeWrite, this was 10 phrases, then each letter 3 times, then 2 more phrases, which took about 15 minutes.

Admittedly, practice for EdgeWrite was more extensive than for the selection-based methods. We acknowledge that the selection-based methods were easier to learn than EdgeWrite. Our goal in EdgeWrite was not to create a more learnable method, but to create a method that offered higher speeds with minimal amounts of practice. Furthermore, subjects quickly became bored with the selection-based methods; requiring equal practice among the techniques would have caused undue fatigue. We include results for users highly practiced in all 3 techniques (Table 3), which show that more practice with the selection-based methods does not result in noticeably improved performance. We also include a graph of speed over tasks (Figure 8), which shows no concerning speedup.

Testing consisted of a fixed set of 10 phrases with each method. Phrase set assignment was even across entry methods to prevent bias. Subjects were instructed to proceed "quickly and accurately" while testing [21].

6.4 Task Phrases

A "task" consisted of entering a single phrase. Our phrases came from [16]. While the practice phrases were chosen at random from a set of 500, test phrases were fixed in sets of 10 and assigned evenly to each entry method. Table 1 shows phrase set characteristics.

Consistent with the reasoning in [16], we did not test numbers, although we did implement them for each method. We believe numbers are common in real-world text entry and should be present even if untested.

Set	Phrases	Words	Chars	Correlation with English
1	10	61	297	89.9%
2	10	52	298	92.7%
3	10	55	298	86.8%

Table 1. Characteristics of test phrases used in the experiment, computed with tools from [16].

6.5 Measures

Quantitative data was logged by the test software and then analyzed according to the measures in [21]. These measures included speed in words per minute (WPM) and accuracy as corrected, uncorrected, and total error rates. In addition, we measured raw data rates in bytes per second (BPS) and logged joystick movements. We obtained subjective data through the use of a post-test questionnaire.

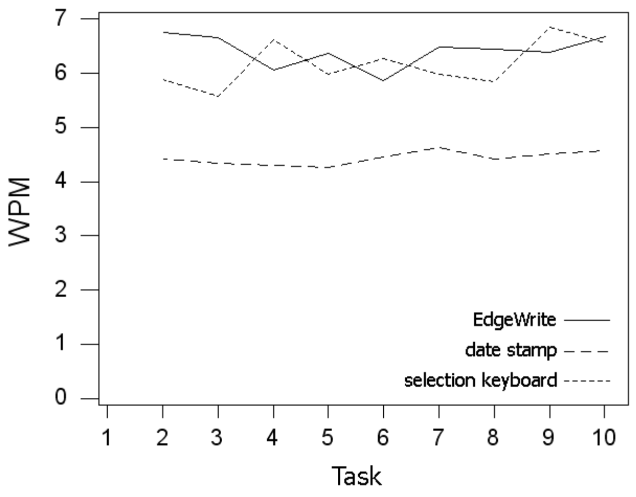


Figure 8. Average words per minute for each method across tasks. Note that only tasks 2-10 are analyzed due to learning in task 1.

7 Results

The data were analyzed using a single-factor within-subjects mixed model ANOVA with a fixed factor for *entry method* and a random factor for *subject*. Contrast tests between tasks 1-5 and tasks 6-10 for each method's speed showed no significant differences for EdgeWrite and date stamp, suggesting that subjects had somewhat stabilized prior to testing. But this contrast test *did* show a difference for selection keyboard ($F_{1,493}$=8.51, p<.01), suggesting that subjects were still speeding up during testing. Most of this speed-up was on the first task. When we removed task 1 from the analyses, contrast tests no longer showed significant speed-up for any method. Thus, all reported analyses are for tasks 2-10.

7.1 Speed

Speed is calculated as words per minute (WPM). Means and standard deviations for our data are: EdgeWrite 6.40 (1.60), date stamp 4.43 (0.62), and selection keyboard 6.17 (1.18).

A main effects test for WPM is significant ($F_{2,466}$=217.20, p<.01). Contrast tests show EdgeWrite is faster than date stamp ($F_{1,466}$=363.80, p<.01) and selection keyboard ($F_{1,466}$=5.11, p<.025). Selection keyboard is also faster than date stamp ($F_{1,466}$=282.69, p<.01).

Speed is affected by accuracy during entry, because it takes time to correct mistakes. But speed does not subsume errors *remaining* in the transcribed string. For this, we use adjusted WPM, defined as WPM × (1 − uncorrected error rate). Results for adjusted WPM are nearly identical to those for WPM, with EdgeWrite's advantage over the other methods being slightly bigger, since subjects had fewer uncorrected errors with EdgeWrite.

7.2 Error Rates

There are three accuracy measures for unconstrained text entry: error rate *during* entry (corrected errors), error rate of the transcribed phrase (uncorrected errors), and a combined measure. These results are shown in Figure 9.

Main effects are significant for all three error rates (p<.01). Contrast tests show that EdgeWrite has a higher error rate *during* entry than date stamp ($F_{1,466}$=73.61, p<.01) and selection keyboard ($F_{1,466}$=132.16, p<.01). Subjects' transcribed phrases, however, are *more* accurate with EdgeWrite than with selection keyboard ($F_{1,466}$=6.24, p<.02), and nearly so than with date stamp ($F_{1,466}$=3.68, p=.055). This discrepancy is discussed below.

Participant conscientiousness (PC) is a ratio of fixed errors to all errors [21]. A score of 1.0 indicates a subject fixed all errors; a score of 0.0 indicates all errors were left in the transcribed string. Means and standard deviations for PC are: EdgeWrite 0.98 (0.09), date stamp 0.89 (0.28), and selection keyboard 0.92 (0.26). A main effects test for PC is significant ($F_{2,466}$=7.39, p<.01). Contrast tests show EdgeWrite PC is higher than date stamp ($F_{1,466}$=13.91, p<.01) and selection keyboard ($F_{1,466}$=7.15, p<.01). Date stamp is not detectably different from selection keyboard.

Thus, despite making more errors during entry, subjects' transcriptions had fewer errors with EdgeWrite, because subjects were more conscientious in correcting mistakes as they went.

7.3 Data and Recognition Rates

Speed only considers the amount of text in the transcribed string. It is also interesting to consider the amount of data transmitted from the text entry device to the computer, the length of the *input stream*. The input stream includes all entered characters, even those later erased, but not non-recognitions. Since a character is one byte, we can describe "data rate" in bytes per second (BPS). This gives us an idea of how fast users produce characters, regardless of how correct those characters are. Note that BPS differs from characters per second (CPS), which is equivalent to WPM, because all transmitted bytes are counted, not just those remaining in the transcribed string.

Means and standard deviations for BPS are: EdgeWrite 0.66 (0.14), date stamp 0.41 (0.08), and selection keyboard 0.54 (0.10). A main effects test for BPS is significant ($F_{2,466}$=361.03, p<.01). Contrast tests show that EdgeWrite is faster than date stamp ($F_{1,466}$=720.72, p<.01) and selection keyboard ($F_{1,466}$=154.34, p<.01). The selection keyboard data rate is also faster than that of date stamp ($F_{1,466}$=208.02, p<.01).

We can use BPS to compute an upper bound for WPM by assuming all bytes are *correct*. In this case, EdgeWrite speed increases 23.0% from 6.40 to 7.87, date stamp 10.8% from 4.43 to 4.91, and selection keyboard 5.3% from 6.17 to 6.50. Thus, EdgeWrite seems to have more potential than the selection-based methods for faster input when accuracy is improved, as would be the case with more practice.

Non-recognitions sometimes occur with gestural interaction techniques. We can compare the number of EdgeWrite gestures made to the number recognized. The average number of gestures made per task was 42.15 (9.73). The average number of gestures recognized per task was 38.07 (7.92). Thus, about 10.6% of EdgeWrite gestures went unrecognized. If all gestures had been recognized and were correct, EdgeWrite's WPM would be 8.65 (1.75). This rate represents perfect performance given novice speeds.

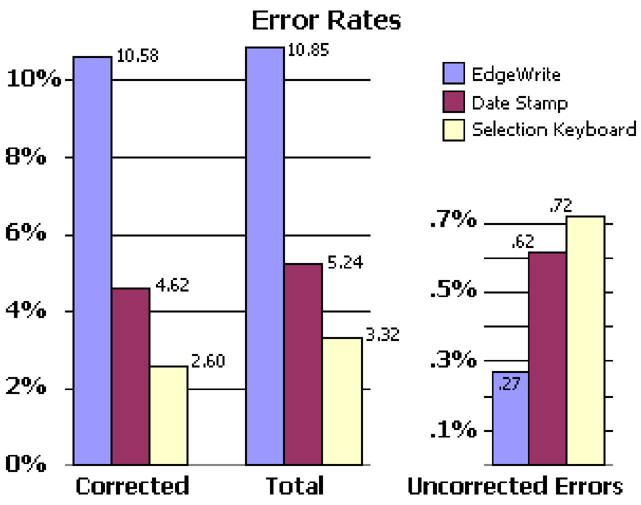


Figure 9. Error rates for unconstrained text entry. Corrected errors are fixed during entry. Uncorrected errors remain in the transcribed string. Total errors is the sum of these.

7.4 Selector Movement

For the selection-based methods, it is interesting to compare the path of selector movement to the minimal selector path. This minimal path is trivial to compute for date stamp: for each letter, spin whichever direction (up or down) reaches the target letter first. For selection keyboard, the minimal path for a phrase can be found using any optimal path-finding algorithm, such as A*.

On average, subjects rotated through 329.81 (64.94) characters per task in date stamp. The minimal path required, on average, 271.77 (46.25) rotations per task. Thus, subjects rotated about 21% more than necessary.

On average, subjects moved the selection keyboard halo 137.51 (25.27) times per task. The minimal path required, on average, 93.19 (13.45) movements per task. Thus, subjects moved the halo about 48% more than necessary.

7.5 Questionnaire Results

We gave subjects four Likert scales (1-5) on which to rate the three entry methods. It is clear from Table 2 that subjects preferred EdgeWrite to the other methods. They felt it was easier, more enjoyable, and faster. Subjects were also less divided in their opinions of EdgeWrite than the other methods, judging by smaller standard deviations.

Likert Scales (1-5)	*EdgeWrite*	*date stamp*	*selection keyboard*
Frustrating-Easy	3.8 (0.6)	3.5 (1.3)	3.2 (1.2)
Painful-Enjoyable	3.9 (0.7)	2.7 (1.0)	2.8 (0.8)
Slow-Fast	3.8 (0.8)	2.7 (1.2)	2.4 (0.9)
Dislike-Like	3.8 (0.9)	2.6 (1.1)	2.6 (1.1)
Average	3.83 (0.75)	2.88 (1.15)	2.75 (1.0)

Table 2. Means (and standard deviations) of post-test Likert scales (ranges 1-5). Labels for scale endpoints are in the left column. Higher values are better.

7.6 Practiced Performance

To see how practiced users fare with EdgeWrite, we tested 3 more subjects, one of whom was an author on this paper. These subjects had prior experience with stylus EdgeWrite and practiced with the joystick text entry methods for 30+ minutes, targeting difficult letters and entering many phrases in each method. The performance of these 3 users with date stamp and selection keyboard was near to that of the subjects in the main study. But with EdgeWrite, these users were clearly faster and less prone to errors (Table 3).

8 Discussion

We should expect a recognition-based method to be less accurate *during* entry than a selection-based method because of misrecognitions. But it is interesting that, despite these errors, novice subjects produced more accurate phrases with EdgeWrite than the other methods, and did so in less time. The selection keyboard requires a second point of visual focus besides the transcribed text, so it is reasonable that subjects may leave errors because they are attending to the keyboard. But on average, EdgeWrite produced more accurate phrases even than date stamp, which requires no secondary focus of attention. Perhaps with EdgeWrite, subjects feel more able to quickly remedy errors, or feel more engaged with their input than with the fairly monotonous selection-based methods. Or, perhaps because of the high error rate during entry, subjects are more vigilant in correcting errors.

Surprisingly, subjects felt that selection keyboard was the slowest of the three methods, even though date stamp was far slower. Subjects also felt selection keyboard was the most frustrating of the methods. Their feedback said this was due to the visual attention required.

In this study, EdgeWrite was faster than selection keyboard by a small margin with many novices over multiple trials. This result should be regarded as the *minimum* amount of practice required by a beginner to become reliably better with EdgeWrite than with selection keyboard. Any further practice, as our results for more practiced users show, only increases EdgeWrite's advantage over the selection-based methods. With even more practice, we expect EdgeWrite's advantage to grow.

Interestingly, the perfect-entry upper bound WPM for our novice subjects (8.65) is still less than the average WPM achieved by our 3 practiced users (10.43). Practiced users made about 36.60 gestures per task compared to 42.15 for novices, a 15% difference.

Practiced Users		1	2	3	*mean*	*novices*
EdgeWrite	wpm	8.57	12.61	10.12	10.43	6.40
	err %	8.11	6.42	4.38	6.30	10.85
date stamp	wpm	4.46	3.69	5.02	4.39	4.43
	err %	3.94	5.59	2.90	4.14	5.24
selection keyboard	wpm	7.12	6.39	6.73	6.75	6.17
	err %	2.38	4.49	2.33	3.07	3.32

Table 3. Speeds and total error rates for 3 practiced users. For comparison, the averages for novices from the main study are shown in the far right column.

9 Future Work

Input methods rely heavily on small details that make big differences [4]. These details must be identified and optimized. For this study, EdgeWrite was not optimized for any physical parameters, as it employed an unmodified off-the-shelf joystick. Over the course of this study, we identified many parameters that could be optimized for better performance in the future.

Some users thought the joystick spring strength was too strong. The height of the joystick should probably be reduced to put the user closer to the underlying control mechanism. The corners of the square bounding area were somewhat rounded, and the joystick sometimes slipped out of them; the corners should be made more abrupt. Subjects felt the size of the plastic square bounding the joystick was too large. This was a particular problem for females, who probably had smaller hands and were, on average, slower and less accurate than males (5.12 *vs.* 6.77 WPM, 12.5% *vs.* 10.4% errors). Many subjects said the joystick's abrasive rubber top made their thumbs sore. Lessons from other input technique development [4] show that large gains are possible with improvements to subtle factors such as these. In fact, after the study was over, we discovered the Logitech Dual Action Gamepad, which has smaller and sharper square bounding areas than the Saitek P2500, resulting in a gain of 1/2 WPM for practiced user #2.

10 Conclusion

The EdgeWrite input technique is well-suited to meet the challenges of joystick text entry because of its Fitts' Law benefits, physical stability, mnemonic characters, and tolerance to wiggle. New users were able to learn EdgeWrite within 15 minutes, after which they could enter text faster and with more accurate results than with date stamp or selection keyboard.

Practiced speeds point to EdgeWrite's potential. Practiced users were 1.5 times faster with EdgeWrite than with selection keyboard, and 2.4 times faster with EdgeWrite than with date stamp. The fastest practiced user wrote at 12.61 WPM, comparable to some stylus Graffiti speeds [20, 27].

Text entry with joysticks does not have to be selection-based. We have shown it is possible and even preferable in some cases to "write" with a joystick.

Acknowledgements

The authors thank D. Gergle, T. Lorusso, H. Seltman, M. Sinclair, and R. W. Soukoreff. This work was funded in part by the NEC Foundation of America, Microsoft Corporation, General Motors, and by the National Science Foundation under Grant No. UA-0308065. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] Card, S.K., English, W.K. and Burr, B.J. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics 21* (1), 1978, pages 601-613.

[2] Coleman, M. *Weegie*. 2001. http://weegie.sourceforge.net/

[3] Co-operwrite, Ltd. *myText*. 1997. http://www.my-text.com/

[4] Ehrlich, K. A conversation with Ted Selker. *interactions 4* (5), 1997, pages 34-47.

[5] Epps, B.W. A comparison of cursor control devices on a graphics editing task. *Proceedings Human Factors Society 31st Annual Meeting*, 1987, pages 442-446.

[6] Goldberg, D. and Richardson, C. Touch-typing with a stylus. *Proceedings INTERCHI 1993*, pages 80-87.

[7] Herz, J.C. *Joystick Nation: How Videogames Ate Our Quarters, Won Our Hearts, and Rewired Our Minds.* Boston: Little, Brown and Co., 1997.

[8] Hirotaka, N. Reassessing current cell phone designs: Using thumb input effectively. *Extended Abstracts CHI 2003*, pages 938-939.

[9] Isokoski, P. Model for unistroke writing time. *Proceedings CHI 2001*, pages 357-364.

[10] Isokoski, P. and Raisamo, R. Device independent text input: A rationale and an example. *Proceedings AVI 2000*, pages 76-83.

[11] Kurniawan, S., King, A., Evans, D.G. and Blenkhorn, P. Design and user evaluation of a joystick-operated full-screen magnifier. *Proceedings CHI 2003*, pages 25-32.

[12] Langolf, G.D., Chaffin, D.B. and Foulke, J.A. An investigation of Fitts' Law using a wide range of movement amplitudes. *Journal of Motor Behavior 8* (2), 1976, pages 113-128.

[13] LoPresti, E.F., Romich, B.A., Hill, K.J. and Spaeth D.M. Evaluation of mouse emulation using the wheelchair joystick. *Proceedings RESNA 2004*, in press.

[14] MacKenzie, I.S. KSPC (keystrokes per character) as a characteristic of text entry techniques. *Mobile HCI 2002*. Berlin: Springer-Verlag, 2002, pages 195-210.

[15] MacKenzie, I.S., Kauppinen, T. and Silfverberg, M. Accuracy measures for evaluating computer pointing devices. *Proceedings CHI 2001*, pages 9-16.

[16] MacKenzie, I.S. and Soukoreff, R.W. Phrase sets for evaluating text entry techniques. *Extended Abstracts CHI 2003*, pages 754-755.

[17] Mithal, A.K. and Douglas, S.A. Differences in movement microstructure of the mouse and the finger-controlled isometric joystick. *Proceedings CHI 1996*, pages 300-307.

[18] Murata, A. An experimental evaluation of a mouse, joystick, joycard, lightpen, trackball, and touchscreen for pointing: Basic study on human interface design. *Proceedings 4th Int'l Conference on Human-Computer Interaction*. Elsevier, 1991, pages 123-127.

[19] Prentke Romich Company. *WiVik On-Screen Keyboard, version 3.* http://www.wivik.com/

[20] Sears, A. and Arora, R. Data entry for mobile devices: an empirical comparison of novice performance with Jot and Graffiti. *Interacting with Computers 14* (5). Elsevier, 2002, pages 412-433.

[21] Soukoreff, R.W. and MacKenzie, I.S. Metrics for text entry research: An evaluation of MSD and KSPC, and a new unified error metric. *Proceedings CHI 2003*, pages 113-120.

[22] Switch-It, Inc. *Mouse Driver.* http://www.switchit-inc.com/pages/Mouse Controls.html

[23] Tarasewich, P. Evaluation of thumbwheel text entry methods. *Extended Abstracts CHI 2003*, pages 756-757.

[24] What Next Research. *KeyStick, version 2.8.0*. http://users.zipworld.com.au/~kevin/KeyStick.htm

[25] Wobbrock, J.O. The benefits of physical edges in gesture-making: Empirical support for an edge-based unistroke alphabet. *Extended Abstracts CHI 2003*, pages 942-943.

[26] Wobbrock, J.O., Myers, B.A. and Hudson, S.E. Exploring edge-based input techniques for handheld text entry. *Proceedings 3rd Int'l Workshop on Smart Appliances and Wearable Computing (IWSAWC 2003).* In *Proceedings of the 23rd IEEE Conference on Distributed Computing Systems Workshops (ICDCS 2003)*, pages 280-282.

[27] Wobbrock, J.O., Myers, B.A. and Kembel, J.A. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. *Proceedings UIST 2003*, pages 61-70.