

Toolglasses, Marking Menus, and Hotkeys: A Comparison of One and Two-Handed Command Selection Techniques

Daniel L. Odell¹

Richard C. Davis²

Andrew Smith¹

Paul K. Wright¹

¹Berkeley Manufacturing Institute, UC Berkeley
Department of Mechanical Engineering
Berkeley, CA 94720-1740, USA
dano@kingkong.me.berkeley.edu

²Group for User Interface Research, UC Berkeley
Computer Science Division
Berkeley, CA 94720-1776, USA
rcdavis@cs.berkeley.edu

Abstract

This paper introduces a new input technique, bimanual marking menus, and compares its performance with five other techniques: static toolbars, hotkeys, grouped hotkeys, marking menus, and toolglasses. The study builds on previous work by setting the comparison in a commonly encountered task, shape drawing. In this context, grouped hotkeys and bimanual marking menus were found to be the fastest. Subjectively, the most preferred input method was bimanual marking menus. Toolglass performance was unexpectedly slow, which hints at the importance of low-level toolglass implementation choices.

Key words: Bimanual interfaces, two-handed interfaces, toolglass, bimanual marking menus, command selection.

1 Introduction

People naturally use two hands when performing physical operations, but standard computer interfaces make use of only one pointing device. Research in bimanual (or, two-handed) interfaces has shown that the presence of pointing devices in both hands can lead to more natural interaction.

Bimanual interfaces can improve **intuitiveness** by:

- *Enabling more input methods* – allowing the body to make virtual manipulations that more closely match physical manipulations.
- *Enabling more sensory feedback* – such as utilizing body awareness (proprioception).

Bimanual interfaces can improve **efficiency** by:

- *Facilitating parallel input* – enabling multiple simultaneous input streams, thereby reducing overall input time.

Bimanual interfaces can improve **comfort** by:

- *Splitting workloads between two limbs* – reducing the load on a single limb.
- *Providing new body positions and motions* – potentially providing more comfortable input.

Taking advantage of these potential benefits requires careful interface design. Previous studies have shown that poorly designed bimanual interfaces can be inferior to standard one-handed interfaces [9, 14]. This study builds on previous work by comparing traditional and new techniques not previously compared together .

- **One-handed techniques**
Standard Toolbars and Marking Menus
- **Keyboard/Pointer bimanual techniques**
Mapped Hotkeys and Grouped Hotkeys
- **Dual pointing device bimanual techniques**
Toolglasses and Bimanual Marking Menus

Bimanual marking menus are a new technique that allows users to manipulate objects with one hand while issuing commands in parallel with the other. The study compares the six techniques above in the context of a simple shape drawing task. This task was chosen for its similarity to commonly encountered tasks. For all methods, completion time and error rates were recorded, and qualitative feedback was collected.

The following section reviews previous work in bimanual interface metaphors and related quantitative studies. Then, the experimental method is described in depth. After a brief look at the participants and environment, results are presented and discussed. The paper closes with conclusions and future work.

2 Previous Work

Buxton and Myers [5] were among the first to demonstrate the potential of bimanual interfaces. They found that experts of their system performed a navigation/selection task 15% faster than experts using a similar one-handed system. Further, they found that as the percentage of parallel activity (*i.e.*, the time that both hands were moving simultaneously) increased, input speed also increased. Recent research has explored bimanual interfaces in other contexts.

2.1 Bimanual Interface Metaphors

Interface metaphors can guide the design of virtual interfaces that more closely map to physical manipulations. The pointer is so well established that few de-

signers pause to think of the interface metaphors it uses. Metaphors for bimanual interfaces, however, are still evolving. This section describes commonly used bimanual metaphors.

Independent hands

A common metaphor treats the pointer as the user's hand, capable of pointing at and "touching" objects on the screen. Thus, a simple bimanual interface metaphor might treat a pair of pointers as two independent hands. This approach fully enables parallel input, while maintaining maximum versatility for each hand. However, a study by Kabbash, et al. [9] demonstrated that two completely independent cursors can lead to reduced performance. They found that cognitive load of the interface increased by splitting users' attention between two separate areas of the display.

Nevertheless, independent manipulations show potential in some applications [14]. For example, one area that shows promise is shape editing [8]. With two-handed "stretchable" shapes, users have much more control of size, form, and position. This can be especially helpful for splines, which have multiple, non-intuitive control handles. This approach has also been extended to 3D volume manipulation using two manipulators with six degrees of freedom [12].

The Kinematic Chain

In 1987, Yves Guiard presented the Kinematic Chain Theory [8], a theoretical framework regarding the way that humans use their two hands. If the right hand is dominant, the basic tenets of the Kinematic Chain Theory are as follows:

Right-to-left spatial reference - the left hand sets the frame of reference for the action of the right hand.

Left-hand precedence in action - the sequence of motion should be left followed by right.

Left-Right scale differentiation - the granularity of action of the left hand is coarser than the right.

For the general case, "left hand" and "right hand" are changed to "non-dominant hand" (NDH) and "dominant hand," (DH). Guiard performed several experiments to support his theory. For example, Guiard observed that people tend to use their non-dominant hand to position paper during handwriting. He found that handwriting speed was reduced by 20% when users were not allowed to use their non-dominant hand to reposition the paper during writing. In the Kinematic Chain Metaphor, then, the NDH sets the frame of reference for the dominant hand. Several studies since then have supported the benefits of using this metaphor in bimanual interfaces [2,9,10].

Parameter and Command

Another bimanual interface metaphor that has received less attention is that of Parameter/Command. The idea behind this approach is that one hand is used to manipulate parameters, while the other is used for command selection [3]. This is similar to how humans perform many common physical manipulations. While stapling, for example, one hand positions the paper stack, while the other hand selects the "staple" operation.

Many aspects of the Parameter/Command metaphor are compatible with that of the Kinematic Chain. In particular, the hand used for command selection can provide a coarser motion, and set the frame of reference (in this case, the mode of operation). These approaches have been successfully coupled (*e.g.*, in Toolglass [2]).

Also the Parameter/Command metaphor is the basis for the most widely used form of bimanual interfaces: hotkeys. In a recent study by McLoone *et al.* [13] both dedicated and chorded hotkeys were faster than the pointer/toolbar interface. Since hotkeys provide increased input parallelism and reduce the need for pointer motions, this result is not surprising.

While few studies have explicitly acknowledged the Parameter/Command bimanual metaphor, several have implemented it in command selection experiments [1,6,9] and demonstrated its benefits. As it seems to be one of the less explored avenues for bimanual computer interfaces, this metaphor was selected for further study.

2.2 Previous Command Selection Studies

The study presented here roughly follows the pattern set by two previous studies. Dillon *et al.* [6] compared toolbar selection speed and error rates when using one mouse and two mice. Users selected either a blue line or a red line from a toolbar to perform a connect-the-dots task. The results showed two-handed command selection techniques to be slightly faster than the one-handed technique.

Kabbash *et al.* [9] conducted a similar study, but used a trackball in the non-dominant hand and also tested palette and toolglass techniques. The experimental task was very similar to Dillon's, but included four color choices. Two independent cursors were found to be slightly slower than one cursor. The palette was no faster than the standard, one-handed toolbar, but the toolglass performed about 16% faster than the toolbar.

It is worth noting that many of the benefits of bimanual interfaces come from higher level activities, such as "chunking" command and parameter selection [4]. The benefits of chunking have been well documented [14,10] and should be considered in any real-world implementation. As in the studies above, this study excludes chunking in order to focus on command

selection in particular. Still, both command and parameter selections are used to simulate real-world tasks.

3 Method

The study presented here used two mice and re-tested the two fastest methods from Kabbash's study: toolbars and toolglasses. The present study also compared marking menus, hotkeys, and bimanual marking menus. The experimental task was a shape drawing task, rather than a connect-the-dots task. This task was selected to imitate common graphics tasks.

3.1 Shape Drawing Task

Command selection techniques were compared by leading subjects through a series of shape drawing trials for each technique. Each trial had the following steps:

1. **Pause Screen:** Gives a clear break between trials. A mouse click moves to the next screen.
2. **Homing Screen:** The dominant cursor automatically homes to screen center, and subjects click to continue. This gives experimental control over shape position relative to initial cursor position.
3. **Draw:** Subjects select a command (line, rectangle, or oval) and draw a matching shape on top of a displayed target shape.
4. **Repeat:** Selecting the wrong command or "missing" the target shape records an error. Subjects must repeat until successful, but only one error is recorded.

All shapes were drawn by clicking on the initial control point, stretching the shape (with the mouse button depressed) to the second control point, and releasing the mouse button. As shown in Figure 1, drawing an oval is analogous to drawing a rectangle, but with fewer visual cues for alignment (the bounding box is not displayed). A shape control point target tolerance of 2.4% of screen size was used for error calculations.

This task simulated disconnected drawing operations similar to those encountered when modifying a diagram with modern graphics software. Unlike the connect-the-dots task, this task does not exploit spatial or temporal locality between operations. Time was measured from the disappearance of the homing screen to the mouse release on the second control point. If the first attempt to draw a target shape failed then the trial was recorded as an error, and excluded from timing analysis. No distinction was made between command

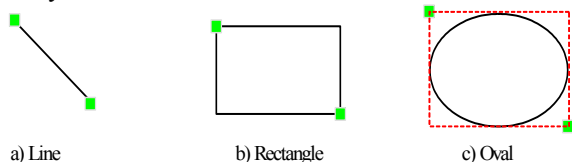


Figure 1: Two control points specify shape, size, and position for all three shapes.

selection errors and parameter specification errors.

Static Toolbars (TB)

The first input method was the standard mouse and static toolbar interface where a single pointer provides all input commands. Using this method, a user moved the pointer from the center of the screen to the static toolbar in the upper-left hand corner of the screen. The user then clicked on the desired shape to draw, and proceeded to draw the shape by clicking on one of the shape's control points, dragging over the shape, and releasing the button to specify the second control point. Since this method is the current standard, it was used as the reference input method.

Hotkeys – Mapped (HKM) and Grouped (HKG)

The second and third methods implemented two flavors of hotkeys: mapped hotkeys and grouped hotkeys. Mapped hotkeys have a cognitive mapping between the key letter and the name of the command that they represent. This mapping may require the user to reposition hands to reach the keys. Key 'R' was used to issue the 'rectangle' command, 'O' for 'oval', and 'L' for 'line.'

Grouped hotkeys have no cognitive mapping between the name of the command and the key letter, but are instead grouped so that they can all be reached without the need to reposition the hand. The number keys '1', '2', and '3' were used for lines, rectangles, and ovals respectively. Both hotkey methods required the user to select the command hotkey, while (in parallel) moving the pointer from the homing position to begin drawing.

Toolglass (TG)

The fourth command method was a toolglass like that presented by Bier, et al [2]. Using this method, the non-dominant hand controlled the position of a toolglass (essentially a see-through movable toolbar) in the workspace. The user then clicked *through* the toolglass (with the dominant hand's pointer), simultaneously specifying the command on the toolglass as well as the first control point. The second control point was then specified as usual.

Standard Marking Menus (MM)

The fifth and sixth modes of input both implemented marking menus. Marking menus are a form of pie menus, which pop up in a radial pattern around the cursor when a button is clicked. With marking menus, the menu does not pop up immediately, allowing commands to be issued quickly from memory without the command menu covering the workspace. After 333 ms, the menu pops up to assist users who have not yet

memorized the command locations. The name “marking menus” refers to commands leaving a marked trail on the screen, giving the user visual feedback.

The test software issues commands when the command cursor crosses the inner boundary of the pie menu, whether the menu has appeared or not. Since no button click is required, this implementation of marking menus also resembles ‘control menus’ [15]. A single button on the dominant mouse was used both for menu selection and shape drawing.

The fifth input mode implemented standard one-handed marking menus. Using this method, the user performed command and drawing operations sequentially. First, the user would click anywhere on the screen and select the desired command as described above. Once the desired command was selected, the shape was drawn.

Bimanual Marking Menus (BMM)

The sixth mode was an experimental mode following the Parameter/Command metaphor. Bimanual marking menus extend traditional marking menus by allowing the left (or non-dominant) hand to select a command while the right specifies shape control points (not unlike the “marking keys” method [1]). To draw a shape, the user first clicks on a shape control point (the first parameter) with the right-handed cursor, which causes a pie menu to become active for command selection with the left hand. Additionally, the pie menu becomes visible after 333 ms. Once the menu is active, the right hand moves toward the second control point while the left selects the command in parallel, as shown in Figure 2. There were two motivations for taking this design approach over standard marking menus:

1. Issuing commands and selecting control points could be performed in parallel, rather than sequentially. This was expected to improve overall speed.
2. Following the object-command metaphor, one hand controlled only command selection, while the other specified only control points. Since hands were not required to shift between these operations, it was expected that the interface would be more intuitive.

3.2 Experimental Design

The independent variables for this experiment include:

Mode: Static Toolbar (TB), Grouped Hot Keys (HKG), Mapped Hot Keys (HKM), Toolglass (TG), Marking Menu (MM), Bimanual Marking Menu (BMM)

Shape: Line, Rectangle, Oval

Size: 220 pixels on diagonal, vs. 440 pixels on diagonal

Position: 128 pixels vs. 256 pixels from screen center

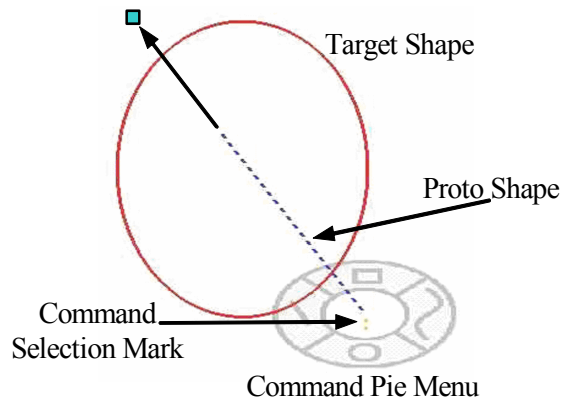


Figure 2: Drawing and selecting the oval command with the bimanual marking menu.

There were 16 repetitions for each condition, broken into eight sequential blocks (of 24 shapes) used to track learning effects as users became more familiar with the input methods. The measured variables were shape completion time, and number of mouse clicks per shape. Using a within-subjects design, the null hypotheses that were tested included:

H₀₁: There was no difference in **average completion time** between input methods.

H₀₂: There was no difference in **learning rate** between input methods. Learning rate was calculated by measuring the improvement in average completion time over the course of the experiment.

H₀₃: There was no difference in **error rate** between input methods. If a shape was not drawn in the minimum number of mouse clicks, an error was recorded, and the error trial’s time was excluded from the completion time analysis.

Testing followed a prescribed order. For each input method, eight blocks of trials were given, in which the shape, size, repetition, and position were randomized. Input methods were presented in randomized order, with the constraint that marking menus and bimanual marking menus were separated by three or more input methods (to reduce cross-learning, as both methods use the same menu). In addition, mapped and grouped hot-keys were separated by two or more input methods to reduce boredom. Note that it was assumed that shape direction from the homing point had no effect on completion time. This assumption was based on the original formulation of Fitts’ Law [7], which accurately predicts that pointing time is solely a function of target distance and size, and ignores direction.

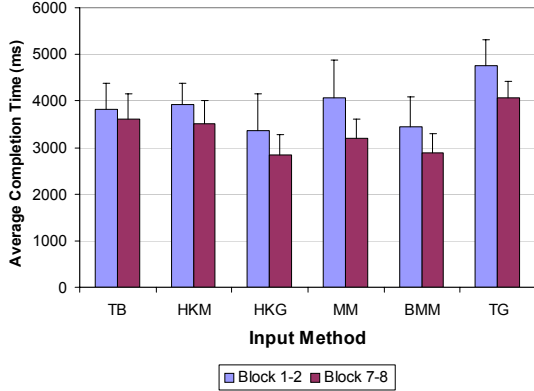


Figure 3: Completion time by block

4 Participants and Environment

All testing was performed with two standard ball mice (one for each hand), and a standard 101-key keyboard. These choices were felt to best represent the standard workstation setup, easily and inexpensively upgraded to enable bimanual capabilities. Right mouse speed was set to 5 out of 11, with ‘low’ acceleration in Win2K. Left mouse speed was controlled by the testing software, and set to a C:D gain of about 1:6, roughly matching the speed of the right mouse. The testing system used a 19” monitor set to 1024x768 resolution.

Twelve volunteer engineering graduate and undergraduate students participated in this study. All had extensive previous computer experience, and used the mouse primarily with their right hand. Eleven participants were male; one was female. Student ages ranged from 21 to 31. A thirteenth participant’s data was discarded due to an extremely high error rate (30% for Toolbar, about 3 times higher than average), and the fact that he completed only 7 of 8 blocks for one of the input modes.

5 Results

The average completion times for blocks 1 and 2^c (novice performance) and blocks 7 and 8 (practiced per-

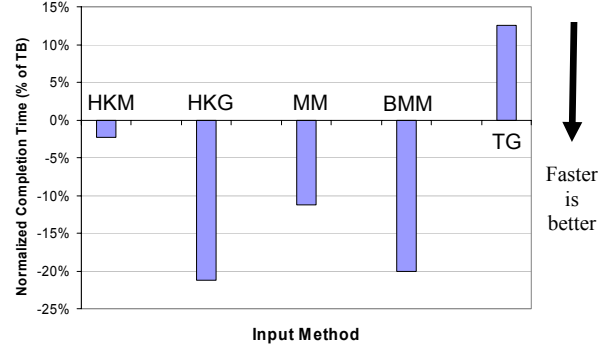


Figure 4: Completion time normalized to Toolbars

formance) are shown in Figure 3. Error bars for all graphs represent standard deviations. As the focus of this study is practiced performance, completion time results are presented for only blocks 7-8 in Table 1. Data were analyzed with one-way ANOVA and follow-up Bonferroni t-tests ($\alpha=.05$) to test the three hypotheses. **H₀₁**: For completion time, the data strongly refute the null hypothesis ($F_{5,3161}=188.98$, $p<.0001$). **H₀₂**: For learning rate, the data refute the null hypothesis ($F_{5,66}=2.76$, $p<.05$). **H₀₃**: For error rate, the data does not refute the null hypothesis ($F_{3,380}=2.51$, $p>.05$). Measured completion times were normalized to the Toolbar completion time (the reference method); the normalized values are plotted in Figure 4.

Error rate was calculated as the percentage of trials for which target shapes were not correctly drawn on the first attempt. Error rates for the hotkey techniques do not appear because they were not captured by the test software. Oval trials accounted for roughly 70% of all errors. This is not surprising, since ovals display fewer cues for alignment, making them more difficult to draw.

Learning rate was calculated as the percent reduction in average completion time between the first two blocks and the final two blocks of the method trial. Figure 5 shows how learning improved average completion time for each device from block 1 to block 8.

At the conclusion of the experiment, participants

Input Method	Toolbar (TB)	Toolglass (TG)	Marking Menu (MM)	Bimanual Marking Menu (BMM)	Hotkeys Mapped (HKM)	Hotkeys Grouped (HKG)
Completion Time (ms)	3616 ^a	4067	3206	2887 ^b	3529 ^a	2846 ^b
Standard Deviation	731	1052	620	726	849	642
Total Error Rate (%)	9.98%	8.29%	10.07%	10.98%	N/A	N/A
Standard Deviation	6.93%	6.06%	7.76%	6.92%	N/A	N/A
Learning Rate (%)	5.53%*	14.11%	19.29%*	15.64%	10.21%	13.51%
Standard Deviation	5.82%	7.33%	14.24%	7.22%	8.62%	12.89%

Table 1: Experimental results. All completion time differences are significant ($p<.05$) except pairs^a and^b, * denotes significant learning difference ($p<.05$)

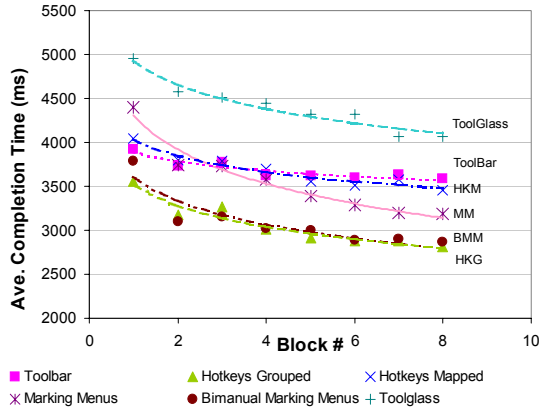


Figure 5: Completion time by block

were asked to rank the six input methods and respond to short-answer questions. Ranking results are shown in Table 2, and differences are significant (Kendall’s $W = .549$, $X^2(5) = 32.95$, $p < .05$). Other user comments appear in the following discussion.

6 Discussion

Of all the tested input methods, Grouped Hotkeys and Bimanual Marking Menus were found to be the fastest. Both were significantly faster than Toolbars, and neither was significantly faster than the other. Toolglasses were found to be the slowest overall. It is interesting to note that the subjective rankings closely follow the quantitative speed rankings.

The performance of Static Toolbars was slightly below average, in terms of completion time, error rate and qualitative ranking. Static Toolbars also demonstrated the smallest learning improvement, due most likely to their ubiquity in today’s interfaces. Overall, these results support our intuition that newer command selection should be considered. The implications of this study for other promising methods are explored further in the following sections.

6.1 Toolglass slower than Static Toolbars?

This implementation of Toolglasses was found to be significantly slower than Static Toolbars. This contradicts the findings of Kabbash et. al. [9], whose study also compared toolbars and toolglasses. The cause of this discrepancy has not been identified conclusively, but there are several possible explanations.

Task differences

This study required users to re-center the pointer and toolglass after every completed shape, and did not display the toolglass until the homing screen disappeared. This was intended to break users’ sense of continuity from shape to shape, and focus the test on command

selection. Since the Toolglass required precise positioning of two onscreen objects, the need to re-acquire these objects before each task may have impaired performance.

To explore this issue further, a smaller follow-up study was performed using six of the same subjects from the previous experiment (considered to be experts with all input methods). This follow-up study used a continuous sequence of 72 randomly located and sized shapes (lines, rectangles and ovals) presented with no homing screen. Only Toolglass and Toolbar input methods were compared. The average completion times were 4.36 seconds for Toolbar ($SD = .73$), and 4.84 seconds for Toolglass ($SD = 1.00$). Thus Toolglasses were still found to be 11% slower than Toolbars, indicating that the presence of the homing screen had only a small effect, if any, on performance.

However, other drawing task differences may also have played a role. Possibilities include drawing colored lines versus shapes, or large target endpoint presentation versus no target endpoint presentation. Additionally, the transparency level of the toolglass may have had an effect.

Input device differences

The pointing device used in this study was the standard ball mouse, the most widely used pointing device. Kabbash’s study, however, used a trackball in the left hand [9]. Also, certain implementation choices differ from Kabbash’s and may affect Toolglass performance.

1. Acceleration - This study had low acceleration for the right hand, while Kabbash had no acceleration. This may have disrupted limb motion synchrony.
2. Control:Display Gains - The ratio of controller movement to pointer movement may have affected Toolglass performance.

Toolbar/Toolglass size

For this study and the Kabbash study, the *relative* size of the Toolbar and Toolglass were identical. However,

Input Method	Average Score	Median Score	Variance
Bimanual Marking Menu	5.25	6	1.48
Hotkeys Grouped	4.58	5	1.90
Marking Menus	3.67	4	1.70
Toolbar	3.17	3	1.97
Hotkeys Mapped	3.08	3	1.17
ToolGlass	1.25	1	.39

Table 2: Subjective rankings

the *absolute* size of the Toolglass and Toolbar appears to be bigger here than for the Kabbash study (154x115 pixels for both). This may have affected relative input method speed, as, according to Fitts' law, the effect of target width on pointing speed is non-linear.

Toolglass should disappear

Most implementations of toolglasses present the toolglass only when it is available for a valid command selection. In our implementation, the toolglass was always visible, even while shapes were being drawn. This implementation decision may have been partially responsible for user's greatest complaint about the Toolglass – that it was 'distracting to use.' Any toolglass motion while drawing a shape may have been visually distracting. Distraction was cited by nine of the twelve participants in the study as a reason for selecting this input method as their least favorite. Other minor differences between the studies (such as four valid menu selections vs. three valid selections) might also have had an effect. A follow-up study is in process to pinpoint the cause for this difference.

6.2 Evaluation of Hotkeys

Despite the fact that both use keyboards, Mapped and Grouped Hotkey performance was found to be significantly different. Grouped Hotkeys were found to be the fastest overall, and the second most preferred. Mapped Hotkeys, on the other hand, were found to be the third slowest (but still faster than Static Toolbars), and the second least preferred. Grouped Hotkey findings reinforce the findings of a recent study [13], which found that both dedicated and chorded hotkeys performed roughly 15% faster than static toolbars for cutting and pasting operations.

Of course, the present study represents a very simple case - where users had to select between only three keys. Due to finger reach and user memory limitations, it is unlikely that grouped hotkeys could successfully be scaled to more than approximately eight to ten commands. In contrast, Mapped Hotkeys could likely be successfully scaled to encompass very large command sets (albeit with inferior performance).

As might be expected, Grouped Hotkeys (where command keys must be memorized) showed a higher learning rate than Mapped Hotkeys (where command keys have a direct cognitive mapping to the command name). In fact, the Mapped Hotkeys learning rate was second only to the static toolbar method, with which users were already expert before beginning the testing.

The main cause for the slow performance of Mapped Hotkeys seemed to be the time required to look from the display to the keyboard in order to locate and select the desired command, and then look back to the

display. This re-homing time was not present for the Grouped Hotkeys method, where memorized key locations were manipulated by feel.

In the Mapped Hotkey trials, several users were observed to contort their hands in an uncomfortable manner to simultaneously reach the 'r', 'l', and 'o' keys – thus turning Mapped Hotkeys into Grouped Hotkeys. In order to maintain the integrity of the comparison when this problem arose, testers quickly reminded subjects to hold their left hand in the home position (index finger over 'f'). This observation demonstrates user preference for grouped hotkeys. This also serves as a reminder that poorly designed interfaces can lead to physical discomfort for users, potentially leading to Repetitive Strain Injuries.

6.3 Evaluation of Bimanual Marking Menus

The experimental input method of Bimanual Marking Menus showed several advantages relative to the other tested input methods. It was subjectively ranked as the favorite input method overall by the test subjects, and was quantitatively found to be the second fastest overall input method. Also, the learning rate of bimanual marking menus was the second highest overall. It offers several benefits derived from bimanual interfaces, including:

- *reduced workload* on the dominant arm by splitting the input between two arms
- *extra degrees of freedom* available for other uses, such as magic lenses, stretchable shapes, etc.

In addition, it combines these benefits with the benefits available from standard Marking Menus:

- *more available workspace* due to the elimination of static toolbars
- *novice/expert transition path* helps users memorize menu locations [11]
- *no visual split* as the command menu is coupled to the pointer position
- *nested menus* allow scaling to larger command sets
- *no dedicated keys* required in hardware

BMMs operate under the Parameter/Command theory. Their implementation diverges from the Kinematic Chain theory, which states that the left hand should precede the right in the series of actions. Instead, Bimanual Marking Menus begin with a right hand motion first. Consistent with KC theory, however, Bimanual Marking Menus use the left hand only for coarse positioning (a ballistic motion is used for command selection). The left hand also sets the frame-of-reference, albeit in a difference context. In this case, the frame of reference is the commanded mode of operation.

Bimanual Marking Menus also require no state switching of the dominant hand, as do many of the other tested input methods. Since the right hand always specifies positions, and the left always specifies commands, there is no thought required to keep track of the current state of the dominant cursor.

In the test software, a new command selection had to be made for every operation. This would need to be changed for practical implementation so that command selection would persist, rather than requiring a new selection for every operation. Additionally, a method for recovering from incorrect command selection would need to be provided, since, in the current implementation, command selection is only available after the first shape control point is specified.

7 Conclusions and Future Directions

This study has quantified the performance of several new and traditional input methods for a simple drawing task. In addition, qualitative feedback comparing these methods was collected from study participants. The most significant findings were as follows:

1. Grouped hotkeys were found to be the fastest overall input method.
2. Bimanual Marking Menus were found to be the second fastest and most preferred input method, and seem to provide other potential benefits that warrant further study.
3. The performance of the Toolglass input method seems to be sensitive to low-level design decisions. The particular implementation of Toolglass in this study demonstrated an inferior performance relative to previous implementations [9].

Toolglass's reduced performance in this study must be investigated more completely. This paper has offered several ideas as to the cause of this result, and future studies will be aimed at testing these ideas.

Bimanual Marking Menus have shown potential. Future work will apply this technique in other contexts. Additionally, the Bimanual Marking Menu method will be integrated with higher-level bimanual concepts to further expand its capabilities.

Source code for software used in this study available at: <http://kingkong.me.berkeley.edu/html/~dano/index.htm>

Acknowledgements

Special thanks to the Ford Motor Fund for supporting campus research labs and enabling studies like these. Thanks also to BWRC for their support. Thanks to our colleagues in UC Berkeley's BMI and GUIR groups for providing valuable feedback, to everyone who partici-

pated in the testing, and to Paul Young for assistance with the test software. Thanks also to Bill Buxton for helping us to interpret our Toolglass findings.

References

- [1] Balakrishnan, R. and Patel, P. The PadMouse: Facilitating selection and spatial positioning for the non-dominant hand. *CHI '98*, 9-16.
- [2] Bier, E.A., Stone, M., Pier, K., Buxton, W. and DeRose, T. Toolglass and magic lenses: the see-through interface. *SIGGRAPH '93*, 73-80
- [3] Brooks, F. The Mythical Man Month after 20 Years, *The Mythical Man-Month*, Addison-Wesley Publishing, Chapter 19, 1995. 260-264
- [4] Buxton, W. Chunking and Phrasing and the Design of Human-Computer Dialogues, *Proc. of the IFIP World Computer Congress*, 1986. 475-480.
- [5] Buxton, W. and Myers, B. A study in two-handed input. *CHI '86*, 321-326
- [6] Dillon, R. F., Edey, J. D., and Tombaugh, J. W. Measuring the true cost of command selection: Techniques and results. *CHI '90*, 19-25.
- [7] Fitts, P.M. The information capacity of the human motor system in controlling the amplitude of movement, *J. of Exp. Psychology*, 47, 1954. 381-392
- [8] Guiard, Y., Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model, *J. of Motor Behavior*, 19 (4), 1987, 486-517
- [9] Kabbash, P., Buxton, W. and Sellen, A. Two-Handed Input in a Compound Task. *CHI '94*, 417-423
- [10] Kurtenbach, G., Fitzmaurice, G., Baudel, T., and Buxton, W. The design and Evaluation of a GUI Paradigm Based on Tablets, Two-hands, and Transparency. *CHI '97*, 35-42.
- [11] Kurtenbach, G., Sellen, A. and Buxton, W. An empirical evaluation of some articulatory and cognitive aspects of "marking menus." *Human Computer Interaction*, 8(1), (1993). 1-23.
- [12] Llamas, I., Kim, B., Gargus, J., Rossignac, J., Shaw, C. Twister: A Space-Warp Operator for the Two-Handed Editing of 3D Shapes. *SIGGRAPH '03*, 663-68
- [13] McLoone, H., Hinckley, K., Cutrell, E., Bimanual Interaction on the Microsoft Office Keyboard, *INTERACT*, 2003.
- [14] Owen, R., Kurtenbach, G., Fitzmaurice, G., Baudel, T. and Buxton, W. Bimanual Manipulation in a Curve Editing Task. *Alias|Wavefront document* 1998.
- [15] Pook, S., Lecolinet, E., Vaysseix, G., Barillot, E., (2000) Control Menus: Execution and Control in a Single Interactor, *CHI 2000* ext. abstracts, pp. 263-4