On Merging Command Selection and Direct Manipulation

Authors removed for anonymous review

ABSTRACT

We present the results of a study comparing the relative benefits of three command selection techniques that merge command selection and direct manipulation: one two-handed technique, Toolglass [2], and two one-handed techniques, control menus [15] and FlowMenu [6].

Our results show that control menus and FlowMenu are significantly faster than Toolglass. Further analysis suggests that merging command selection and direct manipulation is the key factor in the performance of all three techniques.

Keywords: FlowMenu, control menus, tool palette, Toolglass, empirical studies;

INTRODUCTION

Toolglass, a two-handed interaction technique introduced by Bier [2], is the only command selection technique merging command selection and direct manipulation that has been studied empirically. A study by Kabbash [9] showed that Toolglass provides a significant advantage over the more traditional tool palette for a simple color painting "connect the dots" task.

Because there has not been an equivalent study for a one-handed command selection technique merging command selection and direct manipulation, it has been difficult to understand the relative importance of two factors in Toolglass's improved performance: 1) the use of two hands and 2) the merging of command selection and direct manipulation.

To understand the relative performance benefits resulting from each of these two factors, we adapted Kabbash's experiment [9], using a simple color painting "connect the dots" task to compare the commonly used tool palette, Toolglass, and two recently introduced one-handed techniques that merge command selection and direct manipulation: control menus [15] and FlowMenu [6] (Figure 1). While we could have used other one-handed

Tool palette

Toolglass

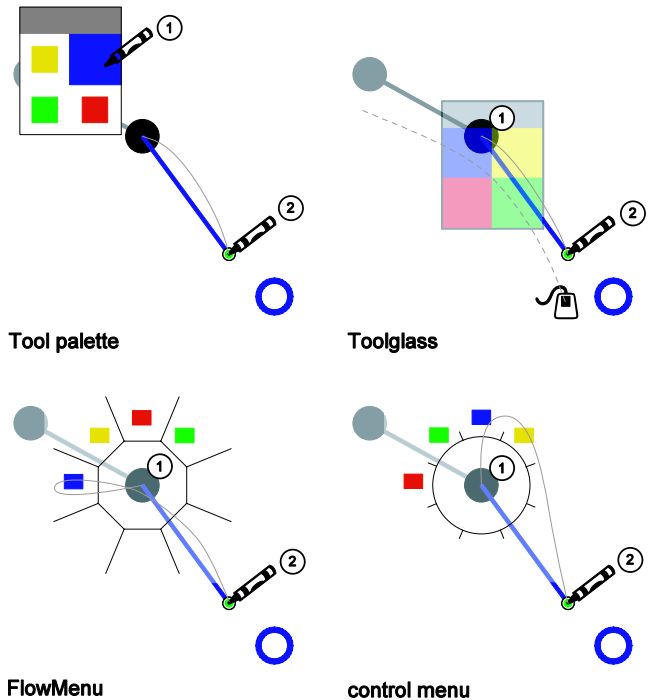**FlowMenu**

control menu

Figure 1: The four command mechanisms used in this study. Each command mechanism is used to select a color before connecting two dots on the screen. Tool palette requires the user to first click on the appropriate color and then connect the dots together. With Toolglass, a two handed technique, users move the semi-transparent Toolglass with a puck and use it to select the color by starting the connection *through* the correct color. Both control menu and FlowMenu are invoked by pressing the pen command button while clicking on the starting dot and performing a gesture to select the color (see text). The path of the pen on the screen during a connection is shown with a light line. The path of the puck is shown with a light dotted line.

techniques that merge command selection and direct manipulation, such as pie menus [7] or the extension of marking menus proposed by Kurtenbach [12], we felt that control menus and FlowMenu were closer to Toolglass's style of interaction. We also judged that these two techniques were different enough that it would be

worthwhile to include both of them in this experiment. Furthermore, neither of them has been studied empirically, adding to the value of our study.

Our experimental results replicated Kabbash's finding that Toolglass is faster than a tool palette for this task and showed that both control menus and FlowMenu are faster than Toolglass. These results seem to imply that the key factor in Toolglass's performance is its merging of command selection and direct manipulation, not two-handedness. Far from questioning the advantages of two-handed techniques, the analysis of our results provides a better understanding of mechanisms at play while using two-handed interactions for command selection. Using our results, we also propose new directions for the design of efficient command mechanisms.

RELATED WORK

Toolglass [2] was one of the first interaction mechanisms to merge command selection and direct manipulation. With Toolglass, the user uses his or her non-dominant hand to manipulate a translucent tool palette and his or her dominant hand to select commands and perform direct manipulation tasks (Figure 1). To perform an action such as creating a colored line, the user first brings the desired color line tool area on top of the starting point and then issues the command by clicking onto the canvas through the Toolglass. The user can then proceed directly with direct manipulation action on the line.

A control menu [15] is a radial menu with 8 octants. Upon activation, the menu pops up, and the user can make a selection by moving from the center toward one of the menu items (Figure 1). When the pen reaches a specified threshold radius, the command is issued and direct manipulation can proceed immediately. Conceptually similar to marking menus [11], control menus use crossing a threshold instead of lifting the pen as a command selection mechanism. This distinction lets the user proceed directly from command selection to direct manipulation without interruption. Like marking menus, control menus can be cascaded to provide access to more than eight commands.

FlowMenu [6] is a radial menu with 8 octants and a central rest area (Figure 1). Upon invocation, the menu pops up centered on the pen. The user selects a top-level menu item by leaving the central area and entering one of the octants. When he or she leaves the rest area, sub-menus for this menu octant appear. Moving the pen to the desired sub-menu octant and reentering the rest area from that octant will trigger a menu selection and direct manipulation can proceed immediately. Although it is similar to control menus in principle, FlowMenu provides additional features like textual entry and knob interaction (see [6] for further details).

Of these three techniques, only Toolglass has been studied extensively. Of particular interest is the experiment conducted by Kabbash that is described in [9]. He tested the performance of different affordances in a simple colored connect-the-dots task. Kabbash compared Toolglass to three other techniques including the conventional tool palette (called *R-tearoff* by Kabbash) and reported significantly better performance in the Toolglass condition, attributing the performance gain to the use of an "asymmetric dependent" [5] technique.

HYPOTHESES

To study how the merging of command selection and direct manipulation influences the performance of interaction techniques, we compared four different interaction techniques:

- Tool palette represents the current *status quo* and neither merges command selection and direct manipulation nor relies on two-handed interaction;

- Toolglass merges command selection and direct manipulation by using two hands to perform the task;

- FlowMenu and control menus merge command selection and direct manipulation with one hand.

We set forth the following hypotheses:

1. The tool palette, which does not merge command selection and direct manipulation will be the slowest condition,

2. Among techniques merging command selection and direct manipulation, Toolglass, a two-handed mechanism, will be faster than both control menus and FlowMenu (following Kabbash's results [9]),

3. A control menu, which has a simpler command selection mechanism, will be faster than FlowMenu.

EXPERIMENT

We closely followed the experimental method used by Kabbash. Subjects were asked to connect a series of colored dots on the screen, using a tool palette, a control menu, FlowMenu, or Toolglass to select a color. While a connect-the-dots task might at first seem artificial, it is in fact quite similar to many interactions in today's interfaces. To make an area selection on a canvas or to create a new object in a CAD program, users often have to first select a tool then perform a drag between two points on the screen. We believe that the task used by Kabbash provides a good abstraction of these everyday tasks but is simple enough to be amenable to accurate measurement.

Like Kabbash, we used a within-subjects design, asking each subject to connect a series of colored dots using all four techniques in turn. The independent variable was the method used to connect dots, and the dependant variable was the total task completion time per connection. Like Kabbash, we believe that total task completion time is a good representation of the overall performance since this

Figure 2: The experimental setting consisting of our display, the Wacom tablet, its pen, and its puck. All conditions used this setting.

measure factors in not only the time taken to perform the task but also the time taken to correct errors.

Subjects

For our study, 12 right-handed, non-colorblind subjects (7 men and 5 women) were recruited from a young adult population (18 to 36 years of age). All subjects had little or no experience using a pen interface other than on a PDA. In addition, subjects had little or no knowledge of control menus, FlowMenu, or Toolglass.

Equipment Apparatus

For our experiment, we used the setup shown in Figure 2. Interactions were performed on a Wacom Intuos 12"x12" tablet. This tablet can simultaneously track a pen and a puck. The tablet was used in absolute mode, and both pen and puck shared the same active surface (unified area setting). Pilot studies showed that the most comfortable setting mapped the screen area to an area 9" by 6 3/4" starting at 3 1/2" below and 3" to the right of the top left corner of the tablet active area. The gain factor between the tablet and the screen was set to 1.33. Finally, to avoid collisions between the pen and the puck, the puck tracking was offset by 1 1/4". This setting was picked for best Toolglass performance according to [1].

The experimental software was running on a Dell Precision workstation 610 MT with a single Pentium III (550 MHz) and 256MB of memory, using an Nvidia GeForce2 as a display card. The workstation was connected to a Dell UltraScan 1000HS series 17" monitor with a visible area of 15" diagonal, running at a resolution of 1024x768. The software logged all the interactions performed by the user. During the experiment, the workstation was disconnected from the network, and logging data was only committed to disk between sets to limit timing errors. To verify the accuracy of our timing method, we also compared the timing provided by our program to the timing provided by
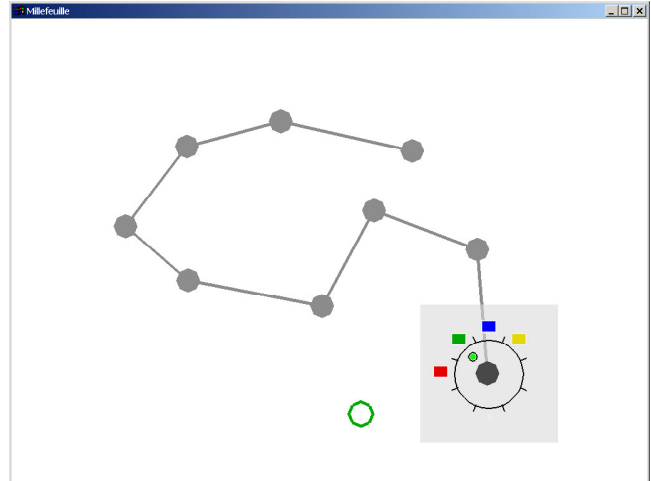

Figure 3: A typical display for our experiment, shown here with the control menu condition. Previously connected dots are shown in gray, while the current dot is shown in black and the target dot is shown open. The 4 possible colors are: red, green, blue, yellow. The gray background has been removed for clarity.

counting fields in video footage of some pilot experiments. In all cases the results were in accord.

Task and setting

For each condition, subjects were presented with 24 sets of 12 points to connect (11 connections per set). For each set, the computer presented the series of colored dots one by one. The subject connected the previous dot to the next dot after selecting the correct dot color using the control mechanism. New dots were presented as soon as the subject successfully connected the active dot, and consecutive dots were always of different colors. The "connection time" was computed from the appearance of a new dot to successful completion of the line, including time to correct any errors in picking the color or connecting the dots. After each set, subjects were presented with their aggregate time for the completed set and their best time so far. If the best time was improved, a rewarding sound was played. Subjects could only rest between sets. All conditions were run with users interacting with a pen (and a puck for the Toolglass) on a digital tablet while looking at a monitor (indirect setting) (Figure 2). Subjects went through the same dot patterns used by Kabbash.

The screen layout is shown in Figure 3. The path created so far is rendered in gray with the exception of the last dot of the path, which is rendered in black. All previous dots in the path are rendered filled. The new target dot is rendered as a circle of the requested color. As soon as a line is started, a line of the selected color is shown on the screen as feedback for the rubber band interaction. This setting, which is slightly different from [9], was used to make the display easier for subjects to parse. Each dot radius was 7/16" and distance between dots varied between 15/16" and 5 15/16".

Tool palette and Toolglass

In the tool palette condition, the color tool palette consisted of 4 buttons, each 5/8" by 5/8", with a header 1 1/4" wide and 5/16" tall at the top, which the subject could use as a handle to move the tool palette. To perform a connection using the tool palette, the subject had to first select the correct color by clicking on the appropriate color button and then had to click on the last dot of the path and perform a rubber band interaction to connect this dot to the new colored target dot.

In the Toolglass condition, the color Toolglass consisted of 4 buttons, each 5/8" by 5/8", with a header 1 1/4" wide and 5/16" tall at the top. The Toolglass was set to 40% transparency so that the dots were visible underneath it. The Toolglass could be moved with a puck. To perform the task using Toolglass, the subject had to first bring the correct Toolglass color area on top of the last dot in the path using the puck. Then he or she had to click on the last dot of the path with the pen and then proceed directly with the rubber band interaction to connect the new colored dot.

In both of these conditions, dots were successfully connected if the pen was lifted from the tablet on top of the target dot.

FlowMenu and the Control menu

In the control menu and FlowMenu conditions, the radius of the menu was 1 3/16". To perform the task using a control menu or FlowMenu, the subject had to first invoke the menu on top of the last dot of the path by pressing the pen's command button while pointing to the dot (the command button could be released as soon as the menu appeared). Then he or she had to select a color by either leaving the rest area through the appropriate color's octant (control menu), or leaving the rest area through the appropriate color's octant then reentering the rest area (FlowMenu). Finally he or she had to proceed directly with the rubber band interaction to connect to the new colored target dot.

In both of these conditions, dots were successfully connected if the pen was lifted from the tablet on top of the target dot or if the command button was pressed while on top of the target dot to start issuing the next command as described in [6].

Protocol

After a brief description of the experiment, subjects were familiarized with the operation of the testing apparatus. For each condition, the correct way to perform the task was explained to the subject, and each subject was given the opportunity to practice on 5 sets of 12 dots. The order of the experimental conditions for each subject was counterbalanced using a Latin square to limit order effects. Furthermore, the color layouts of the control menu and FlowMenu were arranged in different orders, as were those of Toolglass and the tool palette, in an effort to limit carryover effects.

After completing all trials, subjects completed a questionnaire giving subjective ratings of aspects of each technique on a scale from 1 (worst) to 7 (best) and providing information about their previous experience with similar systems. The total time for the experiment was about 1.5 hours.

RESULTS

As in [9], the first connection in each set was removed from the data. As a result, we recorded 240 connections in each of the four conditions for each user. A box plot showed subject S9 (who used a double click to invoke the control menu) as an outlier for error rate in the control menu condition and subjective enjoyability in the control menu condition, so all S9 data points were removed from those two analyses. The results are shown in Figure 4 and Figure 6 with the numerical data tabulated in Table 1.

A repeated-measures ANOVA determined that means for total completion times were significantly different $(F(3,33) = 73.4, \ p < .0005)$. All pairwise comparison significance levels use a Bonferonni correction for multiple comparisons.

Hypothesis 1

As shown in Figure 4, the tool palette was significantly slower than the control menu, FlowMenu, and Toolglass $(p < .0001)$, so **our results support hypothesis 1**. We found similar but slightly better relative improvement between Toolglass and the standard palette than what Kabbash did (22% compared to the 16% reported in [9]). As expected, the two one-handed techniques performed better than the standard palette. Longer path length seems to be the main reason for the inferior performance of the tool palette. The recorded traces (Figure 7) demonstrate the large difference in path length between the tool palette and the other techniques.

Hypothesis 2

Both FlowMenu and the control menu were significantly faster than Toolglass $(p = .01,$ and $p < .0005,$ respectively), so **our results do not support hypothesis 2**. This result is surprising given the extended body of work supporting the superiority of two-handed techniques over one-handed techniques (see [14] for an overview). Contrary to Kabbash's interpretation, our results suggest that Toolglass's advantage over standard techniques might not come from its two-handedness but from its merging of command selection and direct manipulation. It is important to note that our findings are consistent with Kabbash's results. In [9], Kabbash tested three two-handed techniques but observed a significant performance increase only when the technique merged command selection and direct manipulation (Toolglass). These observations reinforce our belief that merging command selection and direct manipulation is the key aspect of Toolglass's superior performance over the standard tool palette technique.
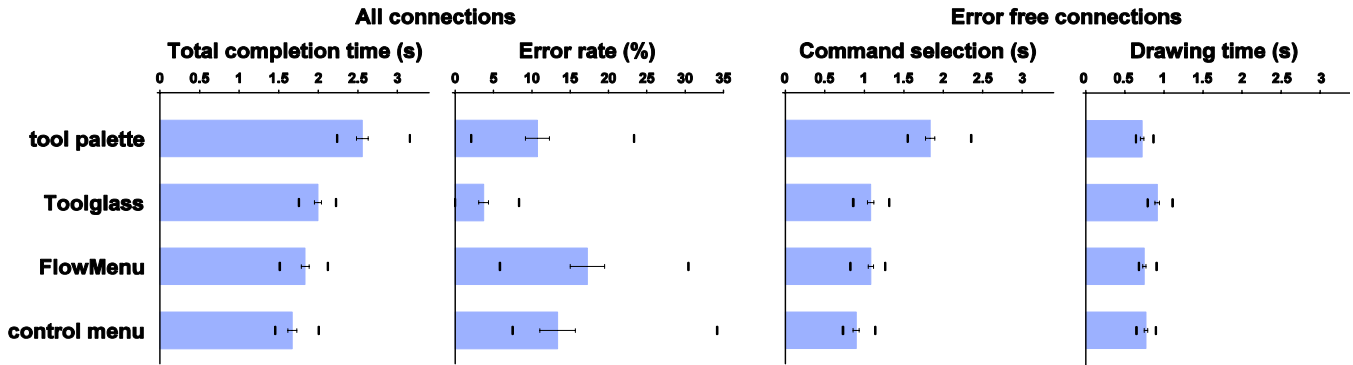
Figure 4: Quantitative results. For each technique, we show from left to right: the average total connection time, the error rate, the average command selection time for connections without errors, and the average drawing time for connections without errors. Besides average values, the standard error, and the maximum and minimum values for each series are shown. Data used to draw these graphs are tabulated in Table 1.

While our main results include errors as part of the connection time, it is interesting to look at connections performed without errors more carefully. In Figure 4, we present the command selection time (time until the color is selected) and the drawing time for error-free connections. In both case, a repeated-measures ANOVA determined that mean times were significantly different, ($F(3,33) = 194$, $p < .0005$ and $F(3,33) = 17.8$, $p < .0005$, respectively). As expected, the tool palette had the slowest command selection time ($p < .0001$), and FlowMenu, which uses a more complex selection gesture, was slower than the control menu ($p < .0005$). Yet, FlowMenu and Toolglass had similar command selection times. At first glance this similarity is surprising since the command gesture for FlowMenu is more complex than the simple combination of moving the Toolglass (the "Prince" law [10]) and pointing. Yet, this similarity can be explained by the fact that both FlowMenu and the control menu only require a gesture without visual feedback; hence, they are not limited by Fitts' law [4].

Toolglass drawing time was slower than that of the other techniques ($p < .005$). To explain this difference, we will focus on the difference between Toolglass and the tool palette, since their drawing tasks are almost identical. We plotted the average connection time for all users against the index of difficulty for our 240 connections. The result is plotted in Figure 5 and shows that connections made with Toolglass and with the tool palette are clustered into two separate populations ($F(2,477) = 2103$, $p < .0005$), with Toolglass connections being more "difficult" to handle. The most commonly observed way users carried out the task with the Toolglass was by moving the Toolglass at the same time as a connection was being made. Thus, it seems that moving the Toolglass with the left hand slows down the tracing part of the task, probably because the user needs to attend to two tasks at the same time.

Hypothesis 3

The control menu condition was faster than the FlowMenu condition, but this difference was not significant ($p = .2$), so

our results do not support hypothesis 3. This result is somewhat surprising given that the control menu command selection time was significantly faster than FlowMenu command selection time for error free connections (see above). We believe that a separate study will be needed to understand this result better.

Error Rates and Subjective Ratings

A repeated-measures ANOVA determined that means for error rates were significantly different ($F(3,30) = 13.2$, $p < .0005$, S9 removed). Toolglass was significantly less error prone than any other technique ($p \leq .01$). One subject was even able to connect 240 points without any errors using Toolglass. No other differences were significant.

Subjective ratings

While a repeated-measures ANOVA determined that means for the "fast" and "error prone" subjective variables were
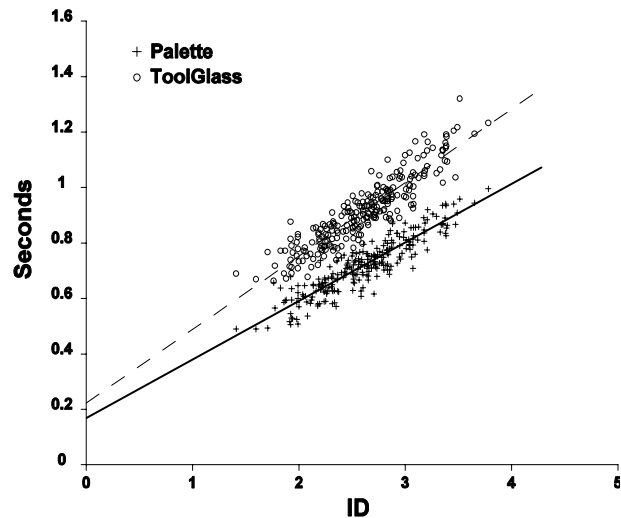

Figure 5: Relationship between the indices of difficulty and the average drawing time (across users) for the 240 connections of our set. ID were computed as: $ID = \log_2(D/S + .5)$ [16].
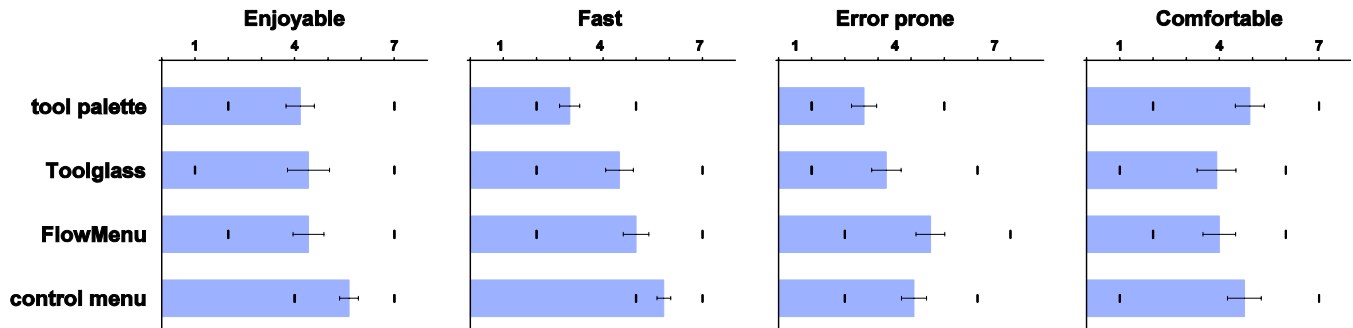
5

Figure 6: Subjective ratings. For each technique we show from left to right: how much users enjoyed using the technique; how fast they felt it was; how error prone they felt it was; and how comfortable they felt it was. Data were collected using a scale from 1 (worst) to 7 (best). Besides average values, the standard error, the maximum and minimum values for each series are shown. Data used to draw this graphs are tabulated in Table 1.

significantly different ($F_{(3,33)} = 11.1$, $p < .0005$, $F_{(3,33)} = 4.8$, $p < .01$, respectively), the only significant pairwise comparisons were found for the fast variable. Overall subjective ratings for these two variables were in accord with the measured speed and error rate. More precisely, both FlowMenu and the control menu were perceived as being faster than the tool palette ($p < .01$), but only the control menu was perceived as being faster than Toolglass ($p < .05$).

We also recorded the users' subjective liking and level of comfort for each technique (Figure 6), but we did not find any significant differences in these cases.

DISCUSSION

The results presented above highlight the importance of techniques that can fluidly mix command selection and direct manipulation. As explained above, such techniques present a significant advantage over the standard tool palette solution in an analog of commonly performed tasks. Our results also suggest that the main advantage of the two-handed Toolglass comes from the fact that it lets the user smoothly merge command selection and direct manipulation.

Interaction design considerations

Our work suggests that control menus and FlowMenu might be faster than Toolglass for tasks like area selection, and vector drawing in CAD and illustration programs. Nevertheless, speed is not the only criterion for designing a user interface. Therefore, it is important to understand some of the fundamental differences between these techniques.

Versatility

Control menus, FlowMenu, and Toolglass might be functionally equivalent in many situations, such as for drawing geometric shapes or for selecting and transforming objects. However, if the application requires freeform drawing, then Toolglass has a distinctive advantage because the Toolglass "see-through" metaphor lets the user start the interaction at the point where the command was invoked. Neither control menus nor FlowMenu provide this flexibility because they require the user to cross a specific

boundary away from the point where the menu was called. Note that pie menus [7] and the extension of marking menus discussed in [12] would face a similar limitation since the selection stroke has to be part of the drawing, or the drawing has to be started at some point away from the initial location where the menu was called.

Scalability

Another important point in selecting command mechanisms is scalability. While it is true that the basic operation in our "connect the dots" task resembles a variety of different direct manipulation actions, many applications have far more than four possible commands. FlowMenu and control menus can easily be extended to accommodate a larger set of commands, but it is not completely clear how one could scale Toolglass without relying on very large and possibly distracting tool or on temporal modes like in T3 [13].

Finally, users often perform similar operations in succession, thereby amortizing the cost of tool selection with the tool palette over several actions. A simple analysis of the data presented here suggests that performing as few as two operations per tool selection might be enough to make the palette faster than Toolglass and FlowMenu (it would take three to be faster than a control menu).

Design considerations

Our results open new avenues in the design of efficient interfaces in situations where two-handed interactions are not practical. For example, for a PDA or tablet computer, it might be difficult to use both hands at the same time or too expensive to install two tracking devices. Another area that might benefit from our finding is that of interaction design for large interactive surfaces. Large interactive surfaces are becoming more and more prevalent and are designed with a direct interaction mode in mind. For those purposes, we believe that a two-handed interface may still have benefits. However, instead of using the non-dominant hand to select the command like in Toolglass, one might use it to set the frame of reference for the work or to manipulate tools like in T3 [13]. Guiard has advocated this setting [5], and our results suggest that this approach might deliver significantly better interfaces.
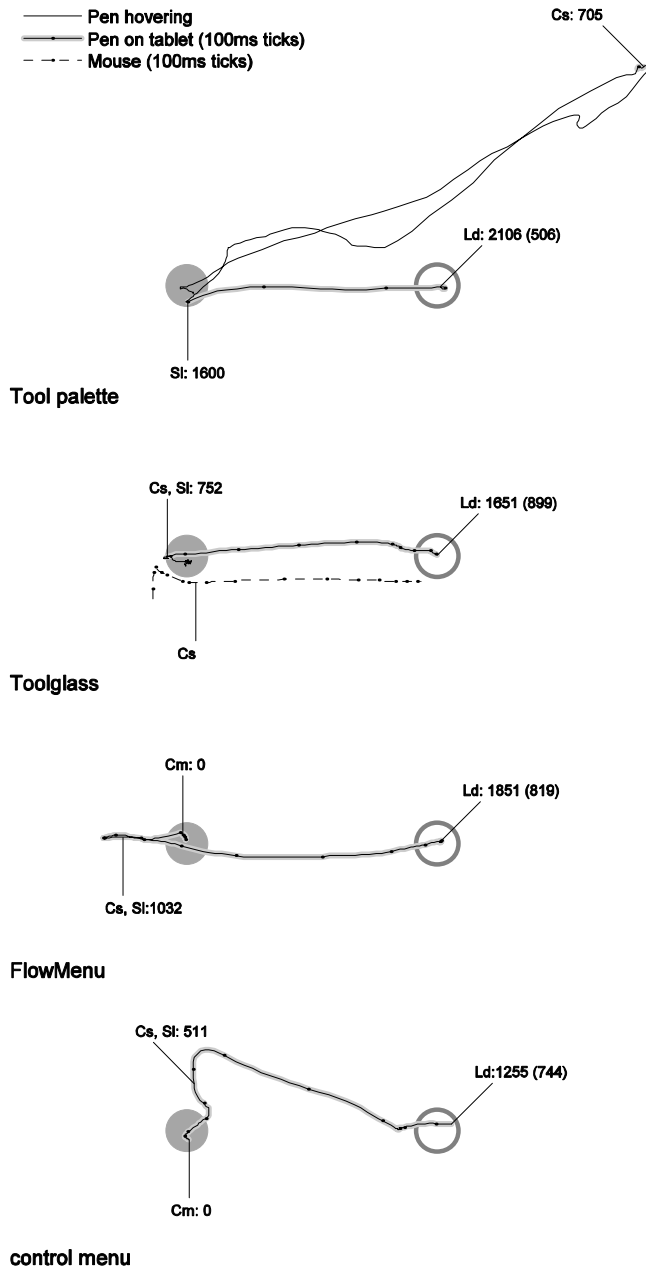
Figure 7: A typical trace for each technique used in this experiment. Reference points show the time (in milliseconds) since the beginning of the connection and are labeled as follows: Cm : Calling menu; Cs: Color selected; Sl: started line; Ld: Line done. For Ld, the time in parentheses is the drawing time.

FUTURE WORK

It is obviously difficult to extrapolate quantitative findings in a controlled experiment to more realistic situations. In our case, we were unable to model how users amortized the cost of command selections with a palette by performing several successive actions with one tool. To solve this problem we are exploring methods to gather quantitative data from real usage behavior for tool palettes in drawing and CAD tools.

We would also like to broaden our understanding of how two-handed techniques can improve other operations. Recent results [3] show the benefits of two-handed interaction for zooming and panning. Using techniques such as speed-dependent zooming [8], we would like to see if similar performance can be obtained using one hand so that the non-dominant hand can be used for other purposes.

Finally, we would like to explore the new design avenues opened by our results. In particular, we would like to pursue new interface designs that either remove the current limitation of control menus and FlowMenu to non-freehand applications or propose a new style of two-handed interaction that lets the non-dominant hand orient the work while the dominant hand performs the work in the frame of reference set by the non-dominant hand.

CONCLUSION

In this paper we have presented new evidence for the benefits of mixing command selection and direct manipulation in commonly performed direct manipulation tasks of modern interfaces. Our results show that these benefits can be obtained not only by using two-handed interaction techniques such as Toolglass but also by using one-handed techniques such as control menus and FlowMenu. Our analysis of these results should aid in the understanding of the advantages of two-handed command selection and in weighing them against possible drawbacks such as more complex hardware. We have also highlighted the need for further analysis to better understand the real advantage of these techniques for actual patterns of use as observed in large industrial-strength applications.

ACKNOWLEDGEMENTS
Removed for anonymous review

REFERENCES
1. Balakrishnan, R., and Hinckley, K. The Role of Kinesthetic Reference Frames in Two-Handed Input Performance, in *Proc. UIST '99* (Asheville NC, November 1999), ACM Press, 171-178.

2. Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. Toolglass and magic lenses: The see-through interface, in *Proc SIGGRAPH '93* (Anaheim CA, August 1993), ACM Press, pp. 73-80.

3. Bourgeois, F. and Guiard Y. Multiscale Pointing: Facilitating Pan-Zoom Coordination, in *Extended Abstracts CHI '02* (Minneapolis MN, April 2002), ACM Press, pp. 758-759.

4. Fitts, P. M. The information capacity of the human motor system in controlling amplitude of movement. *Journal of Experimental Psychology* 47, pp. 381-391.

5. Guiard, Y. Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a

	Tool palette		Toolglass		FlowMenu		Control menu	
Total connection time (s)	2.56	(0.0725)	1.99	(0.0439)	1.83	(0.0486)	1.67	(0.0567)
Error rate (%)	10.7	(1.57)	3.72	(0.64)	17.3	(2.24)	13.4	(2.33)
Command time for error free connections (s)	1.74	(0.0221)	1.02	(0.0276)	0.955	(0.0202)	0.724	(0.0234)
Drawing time for error free connections (s)	0.720	(0.0603)	0.914	(0.0369)	0.747	(0.0311)	0.769	(0.0290)
Enjoyable (1-7)?	4.2	(0.42)	4.4	(0.63)	4.4	(0.47)	5.6	(0.28)
Fast (1-7)?	3	(0.30)	4.5	(0.42)	5	(0.39)	5.8	(0.21)
Error prone (1-7)?	2.6	(0.38)	3.3	(0.45)	4.6	(0.43)	4.1	(0.38)
Comfortable (1-7)?	4.9	(0.43)	3.9	(0.58)	4	(0.49)	4.8	(0.51)

Table 1: Means and standard error (shown in parenthesis) for each variable in our data set

Model. *J. Motor Behaviour*, 19(4) (Dec 1987), pp. 486-517.

6. Guimbretière, F., and Winograd, T. FlowMenu: Combining Command, Text, and Parameter Entry, in *Proc. UIST '00* (San Diego CA, November 2000), ACM Press, pp. 213-216.

7. Hopkins, D. The Design and Implementation of Pie-Menus. 1991. *Dr. Dobb's Journal*, 16(12), pp.16–26.

8. Igarashi, T. and Hinckley, K. Speed-dependent automatic zooming for browsing large documents, in *Proc. UIST '00* (San Diego CA, November 2000), ACM Press, pp. 139-148.

9. Kabbash, P., Buxton, W., and Sellen, A. Two-handed Input in a Compound Task, in *Proc. CHI '94* (Boston MA, April 1994), ACM Press, pp. 417-423.

10. Kabbash, P. & Buxton, W. The "Prince" Technique: Fitts' Law and Selection Using Area Cursors, in *Proc CHI '95* (Denver CO, May 1995), ACM Press/Addison Wesley Longman, 273-279

11. Kurtenbach, G. The Design and Evaluation of MarkingMenus. PhD thesis, University of Toronto, 1993.

12. Kurtenbach, G., and Buxton, W., Integrating mark-up and direct manipulation techniques, in *Proc. UIST '91* (Hilton Head SC, November 1991), ACM Press, pp. 137 - 144.

13. Kurtenbach, G., Fitzmaurice, G., Baudel, T., and Buxton, W. The Design of a GUI Paradigm Based on Tablets, Two-Hands, and Transparency, in *Proc. CHI '97* (Atlanta GA, March 1997), ACM Press, pp. 35-42.

14. Leganchuk, A., Zhai, S., and Buxton, W. Manual and Cognitive Benefits of Two-Handed Input: An Experimental Study. *Transactions on Human-Computer Interaction*, 5(4), pp. 326-359.

15. Pook, S., Lecolinet, E., Vaysseix, G., and Barillot, E. Control Menu: Execution and Control in a Single Interactor, in *Extended Abstracts CHI '00* (The Hague, The Netherlands, April 2000), ACM Press, pp. 263-264.

16. Welford, A. T. Fundamentals of Skill. London, Methuen, 1971.