PieCursor: Merging Pointing and Command Selection for Rapid In-place Tool Switching

George Fitzmaurice, Justin Matejka, Azam Khan, Michael Glueck, Gordon Kurtenbach

Autodesk Research
210 King St. East, Toronto, Ontario, Canada, M5A 1J7
{firstname.lastname}@autodesk.com

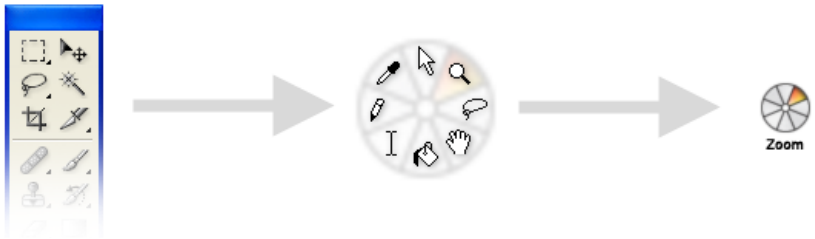

Figure 1. The PieCursor concept: a collection of tools arranged in a radial pattern Tracking Menu and shrunk to the size of a cursor.

ABSTRACT

We describe a new type of graphical user interface widget called the "PieCursor." The PieCursor is based on the Tracking Menu technique and consists of a radial cluster of command wedges, is roughly the size of a cursor, and replaces the traditional cursor. The PieCursor technique merges the normal cursor function of pointing with command selection into a single action. A controlled experiment was conducted to compare the performance of rapid command and target selection using the PieCursor against larger versions of Tracking Menus and a status quo Toolbar configuration. Results indicate that for small clusters of tools (4 and 8 command wedges) the PieCursor can outperform the toolbar by 20.8% for coarse pointing. For fine pointing, the performance of the PieCursor degrades approximately to the performance found for the Toolbar condition.

Author Keywords

Tracking menus, radial menus, marking menus, pie menus, control menus, flow menus, pen based user interfaces, floating palette, multifunction cursor.

ACM Classification Keywords

H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUI), 3D graphics.

INTRODUCTION

Much research has been conducted on finding efficient ways of selecting commands. Toolbars, floating palettes, hotkeys, command lines and pop-up menus all offer ways of switching amongst a collection of tools. Since a user's focus of attention is typically at the cursor location, our investigation explores the potential benefits of adapting the cursor to both point and switch commands on the fly.

Movement of the cursor is performed for a variety of purposes: precision pixel pointing; coarse pointing to select an object; movement to prepare for a subsequent gesture, manipulation and as a user shifts their visual focus in anticipation of action; using the cursor as a visual aid or placeholder when inspecting data; etc. In addition, many commands do not require any spatial information from the cursor (e.g. undo), or only require relative movement during the drag phase as a command parameter (e.g. panning a 2D image).

Given these observations, we investigate opportunities where the cursor could both point and select tools at the same time for the benefit of rapid in-place command selection and operation. In this paper we introduce the PieCursor and report on an exploratory study, a formal performance experiment on the PieCursor, and a set of usability studies which characterize and show advantages of the PieCursor over traditional toolbar workflows.

PIECURSOR TECHNIQUE

A PieCursor is a small (e.g., 32 by 32 pixel) semi-transparent graphical user interface widget that can be controlled by a mouse, pen, or touch screen input device. The PieCursor technique is based on a Tracking Menu [8] design which combines both pointing and selecting a command at the same time.

Tracking Menus use a click-through paradigm where an arrow cursor or some tracking symbol moves within a larger mobile semi-transparent menu of graphical buttons. However, unlike traditional menus, when the cursor crosses the exterior edge of the menu, the menu is moved to keep it under the cursor. The cursor can also be moved within the menu to highlight items. Clicking on an item both selects the item and "clicks-through" to provide pointing to the data beneath the menu.

With the PieCursor, we miniaturize a Tracking Menu down to cursor-size with a radial layout (based on a PieMenu [7, 16] design) to arrange commands (see Figure 1). The cursor is hidden and, in effect, the Tracking Menu itself serves to show cursor location. When the mouse is moved the PieCursor moves. However, depending on which direction the mouse is moved, the command in that direction within the pie layout is highlighted. In our design, command wedges are color-coded with one wedge active at all times. A small label is shown below the PieCursor to indicate the tool name (see Figure 2).

A mouse down event will activate a highlighted command and the PieCursor is replaced by the appropriate tool cursor. A user can then drag to operate the command (e.g. dragging to zoom in/out using the standard zoom tool cursor). Releasing the mouse button brings back the PieCursor (see Figure 3). Thus the selection and operation of any of the commands in a PieCursor is only one button press "away" and can be performed "in-place". Furthermore, the one button press indicates both which command is being activated and the starting point for that command, thus merging pointing and command selection.

Figure 3 shows an example usage of pointing, command selection and command operation for the PieCursor. Panel 1 shows the PieCursor motion to highlight a command. Panel 2 shows clicking to activate the command –a pan tool. Panel 3 shows dragging to operate the pan tool and panel 4 shows the release of the mouse button to deselect the tool and restore the PieCursor.

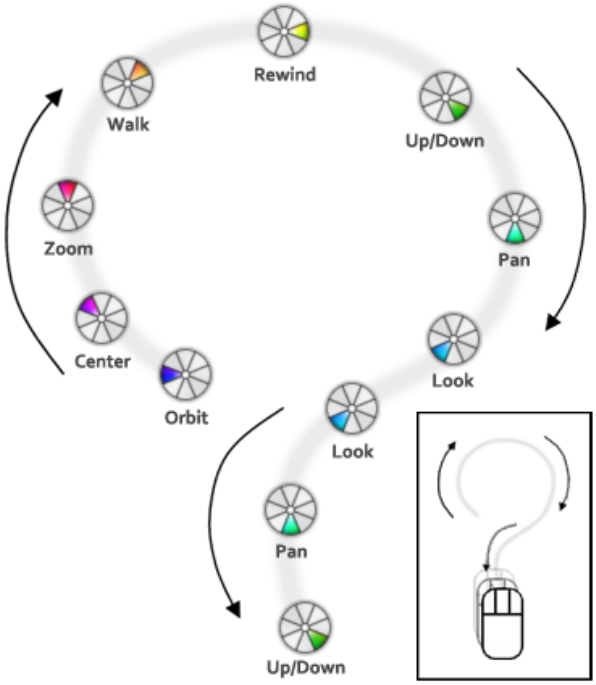

Figure 2. Command wedges chosen by input direction.

To accommodate precision pointing, the user can hit the keyboard Shift key which will lock-in the currently highlighted tool in the PieCursor and replace the PieCursor with an arrow cursor. Holding the Shift key will also retain the current tool for multiple mouse movements and allows the user to repeat a command multiple times until the Shift key is released and the PieCursor returns.

In addition, we have enhanced the PieCursor wedge selection by extending the hit zone of the active wedge (see Figure 4). This allows for a subtle form of wedge stickiness and also serves to reduce the instability that can occur if the input point is at the center of the PieCursor.

The hotspot for the PieCursor lies within the active wedge and is the true cursor position. Since the wedges are fairly

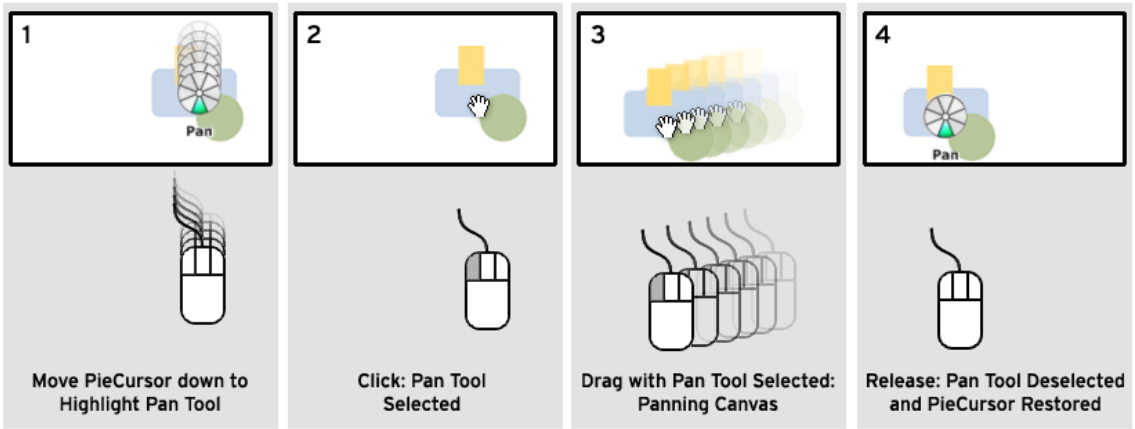

Figure 3. PieCursor usage sequence for highlighting, activating, operating and releasing a command.

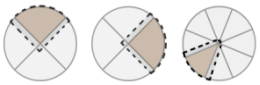

Figure 4. Extended wedge regions (dotted lines).

small, and the true cursor position is usually at the outer edge of the wedge, the hotspot becomes very predictable.

Finally, the PieCursor is not a pop-up menu; rather, like a regular tool, you select it from the toolbar and it becomes your cursor thus changing your input mode.

RELATED WORK

Existing commercial GUIs have several solutions for in-place rapid tool switching. Keyboard hotkeys offer a solution, as they can be used without moving the cursor, but require the use of the non-dominant hand and an available key. Also, a hotkey can be difficult to acquire and articulate, sometimes requiring the user to look down at the keyboard. For a UI designer of commercial applications, a common problem is finding an appropriate hotkey that is not already assigned. The PieCursor differs by not requiring a keyboard (except for precise pointing).

Pop-up menus are a common solution to issue commands at the cursor location but require an explicit input event (e.g., a right mouse button press) to display the menu and present a list of menu items. The PieCursor differs in that no explicit input event is needed to display the menu of choices and to highlight an item.

The PieCursor is intentionally designed to be similar to the traditional GUI modal tool. The design intent is that a PieCursor works "just like a tool" — the user is in a tool mode and presses and drags to apply that tool to the displayed data. However, the tracking PieCursor allows the user to select from several tool modes as they press down. In effect, it is an attempt to cheat and have several tools available within one press-and-drag cycle. Figure 5 shows that using a toolbar often requires 3 steps (selecting a command, pointing at a target and operating the command) whereas the PieCursor requires only 2 steps by combining pointing and command selection.

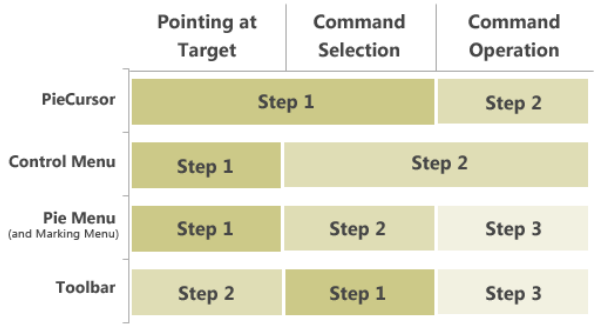

Figure 5. Comparing the "chunking and phrasing" of interaction events across techniques.

In the Microsoft Office 2007 user interface an interaction technique called "mini toolbar" displays a small toolbox close to the cursor after a user completes a drag (after selecting text). While this is an example of a technique to keep tools "close by", PieCursors fundamentally differ by having tool selection occur before the drag action.

A pie menu [7, 16] is a circular popup menu where selection depends on the spatial direction of the target pie slice. The PieCursor builds off of the radial pie menu technique. Our approach differs to be more cursor-like (with rapid in-place tool switching while the input device is in the tracking state) instead of using a pop-up menu paradigm.

A great deal of research has been conducted on radial menus [29] including marking menus [17, 18, 21], pie menus [7, 16], control menus [25], and flow menus [12]. These pop-up menus are typically displayed on mouse down after which a command is selected by dragging. A PieCursor fundamentally differs in that it is a "cursor replacement" where the menu is constantly displayed during the tracking [6] or hover state. The command is selected not during a drag event but during the tracking state by mouse movement. These high level "chunking and phrasing" [5] differences are highlighted in Figure 5.

Figure 6 shows a comparison of the various tool selection techniques using a visualization modified from the KLM style interaction notation of Guimbretiere et al. [11]. For each technique, the input device state and action symbol are shown along with the menu visibility and tool activity with respect to time. The comparison shows, for example, that PieMenus use two input "press" events to select and then perform an action while the PieCursor uses only one.

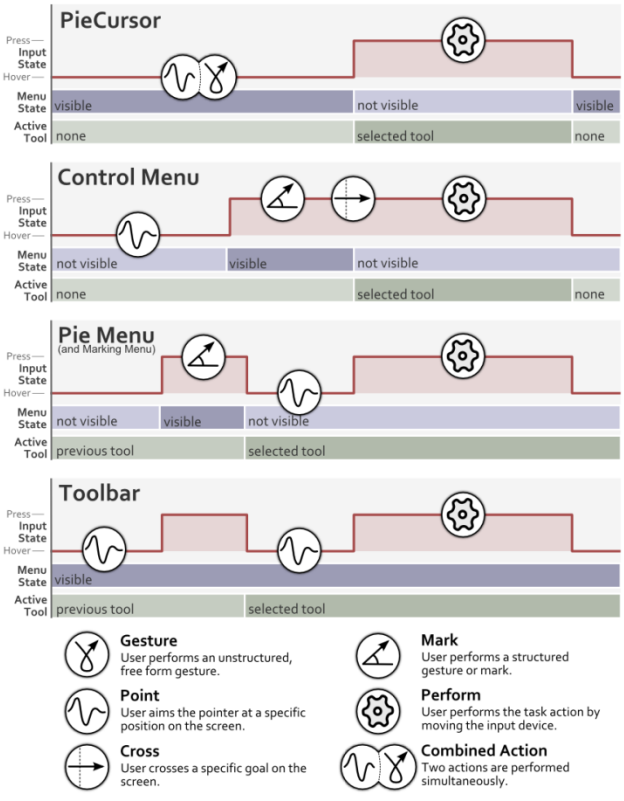

Figure 6. Comparison of interaction techniques.

Multifunctional Cursors [23] examines similar issues as PieCursors – rapid in-place tool switching and function visibility. The major difference is that the Multifunctional Cursor assigns different tools to different mouse buttons whereas the PieCursor only requires one button (and so, is better suited for pen and touch screen configurations).

HoverWidgets [10] is a technique that addresses in-place tool switching as well. The major differences from the PieCursor are that HoverWidgets are designed for pen-operated devices and use short, "L" shape-based gestures in the tracking state to display a clickable widget. The PieCursor uses no gesture recognition and its widget is constantly displayed, acting as a cursor.

Much research on multi-point input also involves in-place, rapid tool switching. Toolglass [4] is another interaction technique that combines tool selection and the starting point of a command. Other work [22] has used finger tracking to perform several operations in place. The major difference is that the PieCursor technique does not require multi-point input technology.

Researchers have studied issues surrounding dynamic cursors [28, 9, 14] and cursor orientations [3, 28]. Findings suggest orientation neutral cursors such as a circle tend to have better performance compared to arrow cursors [24]. As well, research has been conducted on mode switching [26, 20] and inferring modes [27] to stay in the "flow" of interaction [2, 5].

Techniques that use keyboard presses with the non-dominant hand to display widgets near the location of the cursor provide "in-place" tool switching [15, 19]. The major differences are that the PieCursor does not always require a keyboard press and it is a cursor sized widget.

Other research has investigated merging command selection and direct manipulation and studied its benefits [13]. Our PieCursor technique also attempts to realize a benefit of merging but examines the merging of pointing and command selection.

MOTIVATION – REAL-WORLD USAGE

Our motivation for developing the PieCursor results from our work on 3D CAD programs to make navigation easier to learn and use for user's new-to-3D, as well as experts. Typically, 3D CAD applications add to 2D navigation tools to handle the extra dimensions of 3D. For example, 2D zoom and pan tools are augmented with other tools such as "orbit", "walk", "look", and commands such as "reset view" and "center in view". However, in our studies of new-to-3D users, we found that, although this additional functionality was present in the application, many users ignored these additional functions and tried to perform all their 3D navigation tasks with only the 2D tools. We believe that these users did not understand that 3D navigation requires additional commands in many cases. Essentially, users did not recognize that they needed a small cluster of commands, beyond zoom and pan, to properly navigate in 3D. Furthermore, even though the additional 3D tools were placed directly beside the 2D tools on the application's toolbar, once a user had selected a tool, their focus of attention moved away from the toolbar and onto the 3D viewing area. Thus, these other functions for 3D navigation were "out of sight and out of mind". For example, users floundered and became frustrated when they tried to "move to the back of a 3D object" using only zoom and pan.

We reasoned that what was needed was some interaction design that kept these additional tools within the user's visual attention after they moved into the 3D viewing area. One interaction technique which keeps the available tools in the viewing area is Tracking Menus. Thus we developed a Tracking Menu that housed a set of commands which are critical for 3D navigations (Figure 7 front). Subsequent user testing showed that this helped new-to-3D users discover, understand and use the additional commands. Furthermore, other benefits of the Tracking Menu such as reduction of trips to and from the toolbar, and allowing the visual focus to be kept on the 3D viewing area, improved the overall interaction.

However, one side effect reported by some users using the Tracking Menu was that they felt their cursor was "trapped" within the Tracking Menu, (which it is, because that is the way Tracking Menus are defined). With this in mind we began to experiment with making the Tracking Menu smaller until it was the size of a cursor and we could hide the standard cursor. In user testing of these small versions, no users reported the "trapped feeling" and seem to consider the small widget to be their cursor.

Another motivation for using Tracking Menus to access 3D navigation commands was to make interactions faster for experienced 3D users by reducing trips to and from the toolbar to change tools or reduce the need for hotkeys to quickly access navigation functions. We found that among the expert 3D users some preferred and used the large Tracking Menus. Others, however, found the PieCursor version strongly appealing and very useful.

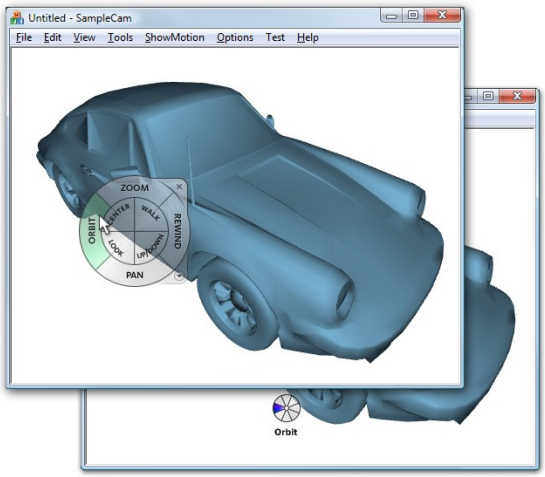

Figure 7. (front) Tracking Menu (back) PieCursor.

EXPLORATORY STUDY

Another large part of our design rationale was based on the assumption that users work and prefer to work "in-place". To get a better understanding of users' cursor placement and operating zones, we conducted an informal usage study to test this assumption.

We created a utility that would constantly monitor and log the mouse cursor position and input state (cursor tracking or mouse down). We then asked three expert 3D modelers to use a professionally available 3D modeling program to construct a sample 3D bottle. No speed or accuracy requirements were communicated. We explained that we just wanted to collect 30 minutes of standard usage data. Figure 8 shows a sample user's cursor activity. The red lines represent mouse dragging activity while the light grey lines represent cursor movement in the tracking state. Green dots on the image represent mouse button presses.

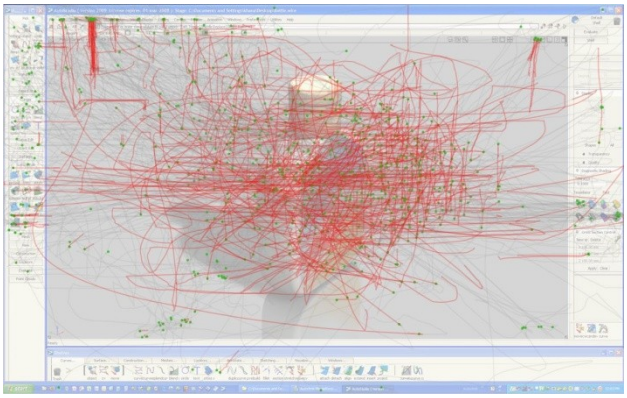

Figure 8. Cursor activity (red = drag; grey = movement, green dot = click).

Our key observation is that all three users showed a concentration of mouse clicking and movement in the center of the application canvas. This reinforces our belief that preference for operating at the center of canvas seems to happen regardless of whether this is efficient in terms of time/motion. For example, some tools can be executed by dragging at any location on the canvas. Thus, the most efficient execution of these tools after selecting them from the toolbar would be to move the cursor just enough distance to get onto the 3D canvas and then drag. However, in observations of everyday usage we do not see this behavior and our study here does not show any evidence of this either. We hypothesize that application data being in the center of the canvas compels users to perform these commands at the center of the canvas.

EXPERIMENT

We conducted a formal experiment to investigate the performance of the PieCursor relative to the larger Tracking Menu and the status quo toolbar. Specifically we were interested in the following questions:

Q1: How does the performance of PieCursor differ from that of the larger Tracking Menu?

Q2: What effect does the number of pie slices in the PieCursor have on performance?

Q3: How does the performance of the PieCursor change between pointing to large and small targets?

Q4: Finally, how does the PieCursor compare to the default base case of simply using tools from the toolbar?

Design

The experiment was conducted using a typical PC workstation configuration (keyboard and mouse) with a 24 inch flat panel screen at 1920x1200 resolution at 100 dots per inch. The experimental design consisted of three interaction methods, two command set sizes and three selection target sizes.

Specifically, three interaction methods were studied: (1) the PieCursor – as described above having a size of 32x32 pixels; (2) the BigWheel – a Tracking Menu having a size of 128x128 pixels with the arrow cursor visible and (3) the Toolbar – a horizontal row of 32x32 adjacent boxes representing tool icons (see Figure 9).

Two command set sizes were examined: four and eight commands. For the PieCursor and BigWheel conditions, four or eight radial wedges were defined whereas the toolbar condition had four or eight tool icons. We also included a "4x4 inner and outer zone layout" version of the BigWheel which was identical to a design we were using in a real application (see Figure 9).

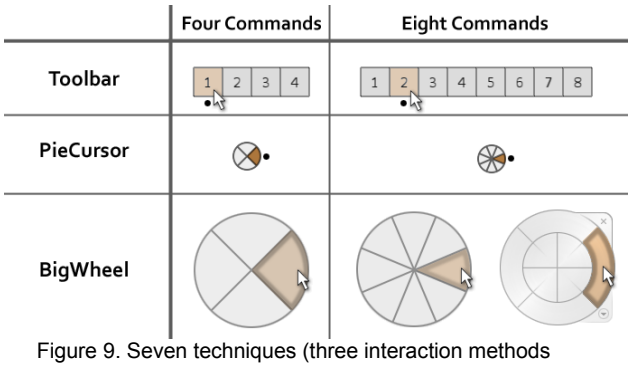

Figure 9. Seven techniques (three interaction methods with 4 or 8 commands).

Three rectangular selection target sizes were examined: small (8x8 pixels); medium (100x100 pixels) and large (1200x600 pixels) based on the exploratory study. The small targets represent precise pointing, for example, to select a control handle or vertex. Medium size targets represent selecting objects. The large target size reflects the working area of the application's canvas.

Task

Our task consisted of selecting a command, clicking on a target and then performing a command parameter by dragging the mouse in a specified distance and direction (see Figure 10). This task represents a common workflow pattern found in a real world task.

Since we were interested in focusing our study on more expert behaviors (usability studies follow which report on first impressions and learning), we designed the task to not require users to memorize command placement. Thus, each trial started with a marker indicating the desired "target command". For the PieCursor and Toolbar condition, we placed a black dot marker next to the target command. For the BigWheel condition we outlined the command wedge in a dark grey color. This was necessary for the 4x4 inner and outer command wedge layouts.

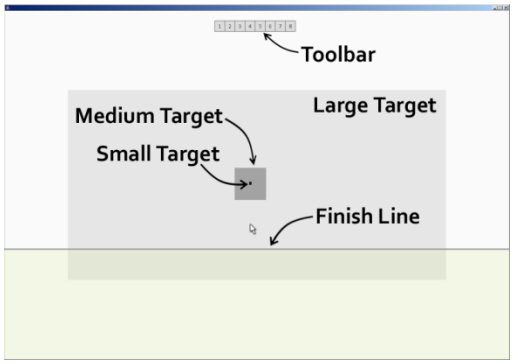

Figure 10. Experiment workspace.

Each trial also had a dark grey box presented to the user in the center of the screen as the selection target. For the PieCursor and BigWheel conditions, users had to position the widget and command wedge over the target and "click-through" to simultaneously select the command and target. For the small target PieCursor conditions, subjects were instructed to use the precision mode. That is, to hit and hold the Shift key to lock in the currently active command wedge which would also replace the PieCursor with an arrow cursor. While continuing to hold the Shift key down, the subject would use the arrow cursor to select the small target. Releasing the Shift key would unlock the command selection.

For the Toolbar condition, a strip of buttons representing icons was centered and placed near the top of the window (535 pixels above the center of the target). Thus for the large target size, a gap of (218 pixels) between the Toolbar and target was present. This was designed to reflect our observations in the exploratory study that users move their cursor towards the center of the canvas before working with the tool. Subjects had to select the tool then move and select the target.

In addition to marking the target command at the beginning of the trial, we also continuously indicated the target drag direction and distance to be performed after the command and selection target was acquired. A vertical or horizontal line across the window was presented and attached to the cursor with a constant distance of 50 pixels (see Figure 10 which shows a downward "Finish Line"). Once the proper command and target were currently selected, the finish line would become stationary during mouse drag events. Subjects were instructed to drag from the target and cross over the line. An ink trail was presented to the user during this dragging for additional feedback. In addition, the target drag zone was colored with a semi-transparent green coloring (see Figure 10). Releasing the mouse in the proper target drag zone (across the finish line) would complete the trial. The next trial started immediately at the subject's current cursor location.

To elucidate more expert behavior, we chose to repeat a sequence of three commands and drag direction pairings for the trials. For all conditions the drag directions followed the sequence: Up, Right, Down. For example, the 4 wedge PieCursor condition had the sequence: North-Up; West-Right; and East-Down. The 8 wedge PieCursor condition had a similar pattern except while we matched all of the requested drag directions we wanted to have one "off-axis" command wedge and thus chose the sequence: North-Up; SouthWest-Right; and East-Down. We tried to match this general command and drag direction pattern with all of the conditions (see Figure 11).

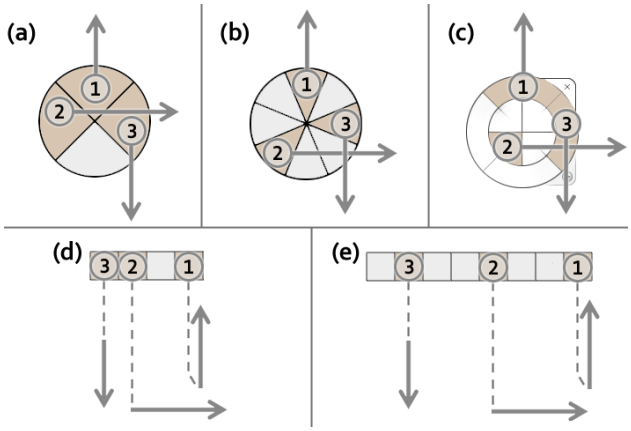

Figure 11. Trial sequences: 1, 2, 3. Command and drag direction shown (a) PieCursor4 and BigWheel4 (b) PieCursor8 and BigWheel8 (c) BigWheel8 4+4, (d) Toolbar4, (e) Toolbar8.

A trial time started when the stimuli were presented to the subject and then finished when they successfully selected the command, target and crossed the finish line (with a mouse up event). Subjects were instructed to work as quickly and accurately as possible. Before each new condition combination of technique, command size and target size, we demonstrated the interaction technique and had subjects practice until they felt comfortable (this lasted from a few seconds to a few minutes). Three blocks of 12 trials were presented for each condition and subjects could rest between blocks.

A total of 12 subjects were used (6 men and 6 women) between the ages of 21 and 38 (all experienced computer users). A within-subjects design was used with each subject using all three methods: Toolbar, PieCursor, and BigWheel. A total of 7 techniques were studied: (1) PieCursor4 – four command size; (2) PieCursor8 – eight command size; (3) BigWheel4; (4) BigWheel8; (5)

BigWheel8 with 4x4 layout; (6) Toolbar4 and (7) Toolbar8. This resulted in 12 subjects x 7 techniques x 3 selection sizes x 3 blocks x 12 trials = 9,072 data points. Trials were grouped by technique and counter balanced with 1/3 of the subjects using the Toolbar conditions first, 1/3 using the PieCursor first and 1/3 using the BigWheels first. The remaining factors were randomly presented.

For every trial, we logged the time for the subject to complete the trial and recorded errors such as missed target, wrong command selected, and we also logged cursor movement. The system waited for a positive match before proceeding to the next trial. After completing the experiment, subjects were given a short questionnaire to determine their preferences for the three methods.

Results

We performed an analysis of variance on the performance data for blocks 2 and 3 (block 1 was dropped to reduce learning effects). A significant difference between the three methods (PieCursor, BigWheel and Toolbar) was found with $F(2, 22) = 141.87$, $p < .0001$. The PieCursor and BigWheel performance was faster than the baseline Toolbar method by 13.5% and 17.5% respectively. No statistical significance in performance was found between the PieCursor and BigWheel methods (see Figure 12).

Thus for Q1: *How does the performance of PieCursor differ from that of the larger Tracking Menu?* We conclude that miniaturizing the Tracking Menu down to the size of a cursor does not significantly degrade performance.

And for Q4: *How does the PieCursor compare to the default base case of simply using tools from the tool bar?* We conclude the PieCursor has significant performance advantage.

We found a significant difference based on whether 4 or 8 commands presented $F(1, 11) = 136.43$, $p < .0001$. That is, it took less time to complete the task with 4 rather than 8 commands. When we group the data based on technique, there is still a significant difference $F(2, 22) = 13.86$; $p < .0001$ (see Figure 13). A Tukey pair-wise T-test of means ($p < .05$) showed no significant differences between the 4 and 8 command Toolbar conditions but statistical differences for the 4 and 8 command PieCursor and BigWheel (4, 8 and 4+4) conditions.

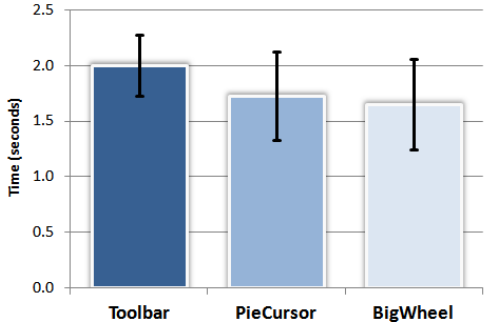

Figure 12. Trial performance mean by method.

Thus for Q2: *What effect does the number of pie slices in the PieCursor have on performance?* We conclude that performance is slightly but significantly degraded with more pie slices.

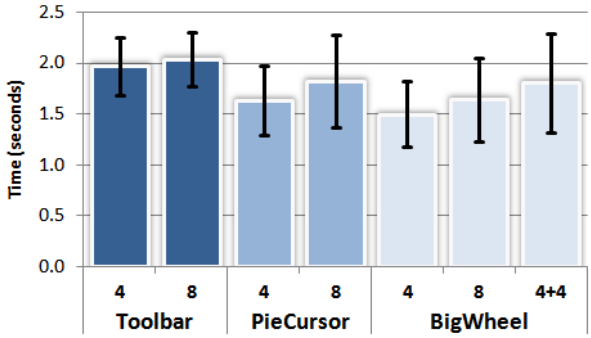

Figure 13. Trial performance mean by technique and number of commands (4 or 8).

Grouping the data by selection target size, we get a significant difference in performance $F(2, 22) = 3281.93$; $p < .0001$ (see Figure 14). A significant interaction was found between target size and method (PieCursor, BigWheel and Toolbar) $F(4, 44) = 82.59$; $p < .0001$. In general, the large selection target condition experienced the best performance for the PieCursor compared to the toolbar with a 20.8% speed improvement. There was a statistically significant performance advantage for the BigWheel compared to the PieCursor and Toolbar conditions for large targets with a 38.6% speed improvement with the BigWheel compared to the Toolbar condition. The larger hit zones of the BigWheels really make a difference at the cost of being a much larger widget than the PieCursor. For medium sized targets the differences between the PieCursor and BigWheel decreases. The PieCursor performs 21.2% faster than the baseline Toolbar method. Lastly, for the small targets (8x8 pixels) the three methods performed roughly the same, as the act of fine precision target pointing dominated the task.

Thus for Q3: *How does the performance of the PieCursor change between pointing to large and small targets?* We conclude that the PieCursor and Tracking Menu advantage does degrade as targets get smaller but getting no worse than traditional "toolbar selection, then point".

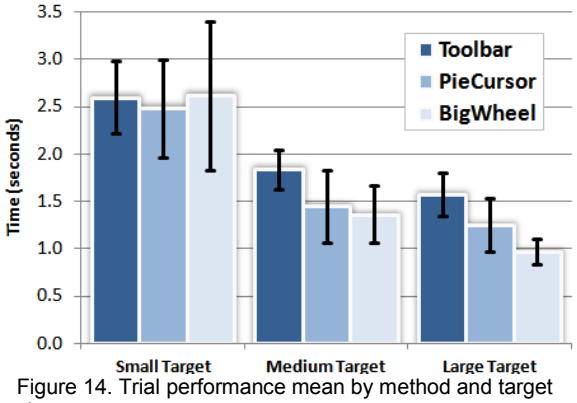

Figure 14. Trial performance mean by method and target size.

In terms of error rate, while significant differences are shown between the conditions, the rates do not greatly differ from the error rates of the toolbar conditions, thus indicating that the different techniques all have acceptable error rate in practice. Means for error rates were significantly different $F(2, 22) = 11.32$; $p < .001$ across the three methods. Examining the error data, we found that the PieCursor4 had similar error rates (7.75%) to that of the Toolbar conditions (8.2%). However, the PieCursor8 had significantly more errors than the Toolbar8 (10.4% vs. 7.1%). Selecting the off-axis command in the PieCursor seemed more challenging. This agrees with other studies on radial menu selection [18] and implies that less frequently used items should be placed in off-axis wedges. The BigWheels had fewer errors than both the Toolbar and PieCursor.

For each trial we logged the cursor distance and found that the Toolbar conditions had significantly more cursor travel (i.e., mouse movement) than both the PieCursor conditions and BigWheel conditions independent of command size and selection target size (see Figure 15). For example, on large targets, 4 times the amount of cursor travel is performed in the toolbar conditions compared to the PieCursor conditions.

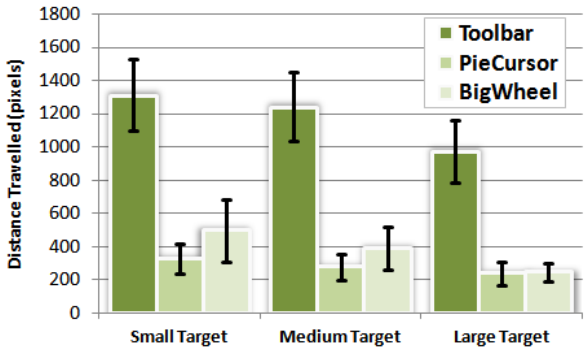

Figure 15. Average mouse travel distances (pixels) per trial, grouped by target size and method.

Overall, consistent with the performance results, subjects indicated that they preferred to work in-place (preferring either the PieCursor or BigWheels over the Toolbar), see Figure 16. However, some subjects commented that the Toolbar conditions seemed easier – since selecting the target command icon felt very familiar and stationary – even though they were doing significantly more mouse movement.

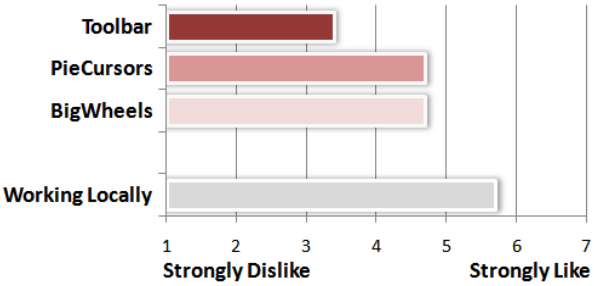

Figure 16. Subjective preference.

GENERAL DISCUSSION

While the experiment reported above shows the PieCursor having performance improvements over the baseline toolbar workflow, the controlled experimental set-up only captures some of the usage situations. For example, in a real-world application context, the user is creating, inspecting or manipulating application data. In the experiment, there was no penalty for looking away from the center of the screen since no application data was presented during the trials. As well, the significant differences of the visual footprint between the BigWheel and PieCursor were not the focus of the study (the interaction footprint was). Expert users often want to minimize the visual interference with the tools and application data.

While the PieCursor and Tracking Menu produced significantly faster task performance times than the toolbar, we were surprised that the toolbar was so fast. This did not seem to match up with our own strong personal preferences for the PieCursor and Tracking Menu over the toolbar for our real work 3D navigation tasks. We hypothesized that subjects in the experiment seem to move much faster that users doing real work and found evidence to support this. In the experiment, subjects moved to the toolbar with an average speed of 593 pixels/sec. In contrast, in our exploratory study of users doing a real modeling task (i.e. Figure 8), users moved to the toolbar at a dramatically (3.6 times) lower average speed of 163 pixels/sec. Similarly, the average speed used in selecting an item with the PieCursor and Tracking Menu in the experiment was 157 pixels/sec. We speculate that this may be evidence that fast operation of toolbars is possible but requires more effort than using the PieCursor or Tracking Menu hence the preference for the latter in a real world task.

The PieCursor offers two styles of command selection: on-site and on-approach. With on-site selection, users are either already at their selection target or roughly move to the target and then make a fine adjustment to the PieCursor to select the desired command. In contrast, on-approach behaviors alter the vector towards the target e.g. a user may approach a target from below and then move up to simultaneously reach the target and select the "north" wedge in the PieCursor.

We have noticed that, in practice, both the on-site and on-approach behaviors for selecting a command with the PieCursor feel very lightweight. Examining the PieCursor interaction design more closely, we have introduced a contrast in tension between the act of tool selection as a "lightweight" activity (i.e. mouse tracking) and command operation as a "stronger" activity (i.e. mouse dragging). This contrast seems to be significant and useful. This design differs from say marking menus which use the same "mouse-down and drag" activity for both command selection and command operation. Control menus also use the same activity (mouse drag) for both command

selection and operation. Removing the tension during command selection allows for easy and rapid switching between a collection of commands.

The tradeoff in the PieCursor design is the challenge of precision pointing and locking-in a command/tool mode. We currently solve this by introducing a keyboard mode (using the Shift key) to lock-in a command/tool and to offer precision pointing by temporarily replacing the PieCursor with an arrow cursor. Future research is needed to determine if alternative designs can address this issue.

Showing the standard arrow cursor within the PieCursor makes a visually heavy and confusing design. We have investigated alternative designs (see Figure 17) such as offering a smaller dot cursor (what we call the "flea") or a miniaturized arrow cursor within the PieCursor that indicates the precise mouse input and active hotspot. Another design we studied was not to show the PieCursor but display a modified arrow cursor. When the arrow cursor moves into a new command wedge from the invisible PieCursor, we change the visible cursor to the corresponding tool cursor. We thought this would work well but due to a lack of predictability, it did not. Yet another approach would use a click-hold design during command activation within the PieCursor technique to lock-in/release a command/tool mode. Lastly a velocity based solution may serve as a means of switching between command selection (high velocity mouse movement) and precision pointing (low velocity mouse movement).

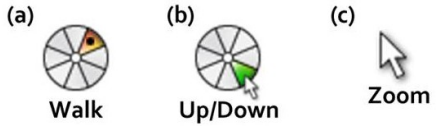

Figure 17. (a) flea cursor; (b) mini-arrow; (c) directional arrow.

Through our development of the PieCursor technique, we have found that it works well for pen based systems. In addition, it performs fairly well for touch-based systems since the user can use the PieCursor as an anchor position and touch in the corresponding virtual wedge that extends beyond the PieCursor to both move the PieCursor and select a command during the initial touchdown state

USABILITY STUDIES

We have also extensively evaluated both Tracking Menus and the PieCursor as part of a project to add these techniques to several commercial 3D CAD applications. User experience observation studies were used where users were asked to perform real 3D navigation tasks and "think aloud" to elucidate their initial impressions and experience with these new widgets. Our first study involved thirty users —a mixture of experienced and inexperienced 3D users— using Tracking Menus. In general, users within a minute or two understood how the technique worked and successfully completed tasks after a few minutes. A small number of users even understood and reported back the benefits of Tracking Menus for "in-place" work over the toolbars. Also from these studies came the comments mentioned earlier from some users that they "felt their cursor was trapped" or "the Tracking Menu was too large".

We subsequently developed the PieCursor to address these issues and tested it, versus the Tracking Menu in similar "think aloud" observational studies with 10 users who were professional users of 3D CAD architectural applications. With this set of users we found that after a minute or so 9 out of 10 users understood and started to use the technique and, after 15 minutes of experience, all users could comfortably use the two techniques for a variety of 3D navigation tasks. Users were split about 50/50 in terms of preference for the PieCursor over the Tracking Menu. No user reported "feeling trapped" in the PieCursor, nor did they report that their "cursor was missing"—it seemed to go without saying that the PieCursor was the cursor. In general, the amount of on-screen feedback in the design of the PieCursor seems sufficient for users to quickly overcome the novelty of the interaction technique.

CONCLUSIONS & FUTURE RESEARCH

We can conclude from our exploratory study, experiment, and usability studies that the PieCursor is a useful and advantageous interaction technique, especially for the application of in-place command selection and operation. Both the PieCursor and Tracking Menu methods outperformed the Toolbar method, especially when the user did not need to always point to very small targets. In effect, the PieCursor does allow a user to have several tools attached to the cursor at the same or lower cost than having one tool from the toolbar.

While we addressed pointing to small targets by the "Shift key" function for the PieCursor, future research would look at ways which do not rely on the keyboard. Other limits of the PieCursor technique such as number of items, and support for persistent tools are also subjects for future design and research.

Other tool and command selection techniques may have obvious performance advantages and comparisons to such techniques, such as hotkeys, are valid but perhaps more interesting for future research are investigations that look at the combination of techniques. In a similar vein, while this paper has shown advantages over toolbars, it is interesting to consider the design implications of having PieCursors as tools on a toolbar, collapsing 4 or 8 toolbar icons down into one thereby reducing toolbar clutter. Moreover, PieCursors may serve as an interesting example for new interaction model explorations such as Instrumental Interaction [11] which could be extended to include the size dimension of the widget.

Additional contributions of this work include evidence of a preference and behavior for working in-place, in the

center of an application, and evidence of large differences in mouse movement speed between experimental laboratory settings and during real world tasks.

Finally, the PieCursor is a new technique which can perform 20% faster than a toolbar. It is a general solution that can be easily added to existing applications.

REFERENCES

1. Beaudouin-Lafon, M. Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces. In Proc. CHI 2000, ACM Press (2000), 446-453.
2. Bederson, B.B. Interfaces for Staying in the Flow. *In Ubiquity 5,* 27*,* ACM Press (2004).
3. Bederson, B.B. LiveCursor – A Jazz Applet, University of Maryland, http://www.cs.umd.edu/ ~bederson/livecursor/
4. Bier, E.A., Stone, M.C., Pier, K., Buxton, W., and Derose, T.D. Toolglass and magic lenses: The see-through interface. In *SIGGRAPH'93.* (1993), 73-80.
5. Buxton, W.A. Chunking and phrasing and the design of human-computer dialogues. In *Human-Computer interaction: Toward the Year 2000*, R.M. Baecker, J. Grudin, W.A. Buxton, and S. Greenberg, Eds. Morgan Kaufmann Publishers, (1995), 494-499.
6. Buxton, W. Three-state model of graphical input. *Interact Conference,* (1990), 449-456.
7. Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. An empirical comparison of pie vs. linear menus. *Proc. CHI 1988*, ACM Press, 95-100.
8. Fitzmaurice, G., Khan, A., Pieke, R., Buxton, B., and Kurtenbach, G. Tracking Menus. In *ACM UIST Symposium*. (2003), 71-79.
9. Grossman, T. and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. CHI 2005*, ACM Press (2005), 281-290.
10. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., and Balakrishnan, B. Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-Operated Devices. In *Proc CHI 2006,* ACM Press (2006), 861-870.
11. Guimbretière, F., Dixon, M. and Hinckley, K. ExperiScope: An Analysis Tool for Interaction Data. In *Proc CHI 2007*, ACM Press (2007), 1333-1342.
12. Guimbretière, F., Stone, M. and Winograd, T. Fluid interaction with high-resolution wall-size displays. In *ACM UIST Symposium*. ACM Press (2001), 21-30.
13. Guimbretière, F. Benefits of Merging Command Selection and Direct Manipulation. In *ACM Transactions on Computer-Human Interaction,* 12, 3, ACM Press (2005), 460-476.
14. Hertzum, M. and Hornbaek, K. Input Techniques that dynamically change their cursor activation area: A comparison of bubble and cell cursors. In *International Journal of Human-Computer Studies*, 65, 10 (2007), 833-851.
15. Hinckley, K., Guimbretiere, F., Baudisch, P., Sarin, R., Agrawala, M., Cutrell, E. The Springboard: Multiple Modes in One Spring-*Loaded Control*. *ACM CHI 2006*. ACM Press (2006), 181-190.
16. Hopkins, D. The design and implementation of pie menus. In *Dr. Dobb's J.* 16, 12 (Dec. 1991), 16-26.
17. Kurtenbach, G. The design and evaluation of marking menus. Ph.D. Thesis. University of Toronto. (1993).
18. Kurtenbach, G., and Buxton, W. The limits of expert performance using hierarchical marking menus. In *ACM CHI'93*. ACM Press (1993), 35-42.
19. Kurtenbach, G., Fitzmaurice, G., Owen, R., and Baudel, T. The Hotbox: Efficient Access to a Large Number of Menu-items. *ACM CHI'99*, 231-237.
20. Li, Y., Hinckley, K., Guan, Z., and Landay, J.A. Experimental analysis of mode switching techniques in pen-based user interfaces. In *Proc. CHI 2005*, ACM Press (2005), 461-470.
21. McGuffin, M.J., Burtnyk, N., and Kurtenbach, G. FaST Sliders: Integrating Marking Menus and the Adjustment of Continuous Values. In *Graphics Interface,* (2002), 35-41.
22. Moscovich, T., and Hughes, J.F., Multi-finger Cursor Techniques, In *Graphics Interface*, (2006), 1-6.
23. Muller, M.J., Multifunctional Cursor for Direct Manipulation User Interfaces, *Proc. CHI '88*, 89-94.
24. Po, B.A., Fisher, B.D. and Booth, K.S. Comparing cursor orientations for mouse, pointer, and pen interaction, *Proc. CHI 2005*, ACM Press, 291-300.
25. Pook, S., Lecolinet, E., Vaysseix, G. and Barillot, E. Control menus: Execution and control in a single interactor. In *ACM CHI 2000,* ACM Press, 263-264.
26. Sellen, A., Kurtenbach, G., and Buxton, W. The Prevention of Mode Errors through Sensory Feedback. In *Journal of Human Computer Interaction*. (1992), 141-164.
27. Saund, E. and Lank, E. Stylus input and editing without prior selection of mode. In *ACM UIST '03 Symposium*, ACM Press (2003), 213-216.
28. Tian, F., Ao, X., Wang, H., Setlur, V., and Dai, G. The tilt cursor: enhancing stimulus-response compatibility by providing 3d orientation cue of pen. In *Proc. CHI2007*, 303-306.
29. Wiseman, N. E., Lemke, H. U., and Hiles, J. O. PIXIE: A New Approach to Graphical Man-machine Communication. *Proc. of 1969 CAD Conference Southhampton*, 463, IEEE Conf. Publication 51.