Tracking Menus

George Fitzmaurice, Azam Khan, Robert Pieké, Bill Buxton, Gordon Kurtenbach

Alias|wavefront
210 King Street East
Toronto, Ontario M5A 1J7, Canada
E-mail: {gf | akhan | rpieke | gordo}@aw.sgi.com; bill@billbuxton.com

ABSTRACT

We describe a new type of graphical user interface widget, known as a "tracking menu." A tracking menu consists of a cluster of graphical buttons, and as with traditional menus, the cursor can be moved within the menu to select and interact with items. However, unlike traditional menus, when the cursor hits the edge of the menu, the menu moves to continue tracking the cursor. Thus, the menu always stays under the cursor and close at hand.

In this paper we define the behavior of tracking menus, show unique affordances of the widget, present a variety of examples, and discuss design characteristics. We examine one tracking menu design in detail, reporting on usability studies and our experience integrating the technique into a commercial application for the Tablet PC. While user interface issues on the Tablet PC, such as preventing round trips to tool palettes with the pen, inspired tracking menus, the design also works well with a standard mouse and keyboard configuration.

KEYWORDS: pen based user interfaces, menu system, graphical user interface, floating palette, Tablet PC.

INTRODUCTION

With the widespread availability of pen-based computers, such as the Tablet PC (see Figure 1) and pen-based PDAs, tablet computing is becoming ubiquitous and mainstream. User interface designers increasingly face the challenge of designing effective pen-based user interfaces for this class of devices.

A primary requirement is to offer rapid switching between different tools on a pen-only system. For example, consider rapidly switching between a drawing tool and a pan tool in a drawing application. This would require repeated trips from the drawing area to the tool palette, selecting the pan tool, then moving back to the drawing area to pan, then returning to the tool palette to re-select the previous tool. This results in a large amount of travel time for the user and becomes quite monotonous in practice. We call this behavior *tool palette round trips*.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UIST '03 Vancouver, BC, Canada
© 2003 ACM 1-58113-636-6/03/0010 $5.00

In keyboard-based systems, alternate ways of switching between tools (keyboard accelerator techniques) are typically provided to reduce travel time. For example, in Adobe Photoshop, a very popular feature is an accelerator technique in which the system switches from the current tool to the pan tool when the user depresses the space bar key. Thus, trips to and from the tool palette are not necessary to use the panning tool.

In pen-only systems, there is no keyboard available and therefore other techniques are required to reduce travel time. A pen-barrel button could be used to switch tools but these buttons are often mistakenly pressed, causing an error, or are very awkward to press. Another option is to use hardware buttons around the bezel of the display but these are also awkward to press while the user is holding the tablet. Moreover, these buttons have fixed system-wide functions assigned by the current Tablet PC standard, making them inappropriate for application-specific commands. Voice recognition, pen gestures, or sensing physical manipulation of the tablet [11] may be other alternatives, but rely on recognition techniques that may impede rapid tool switching and would be more appropriate for less frequent operations.

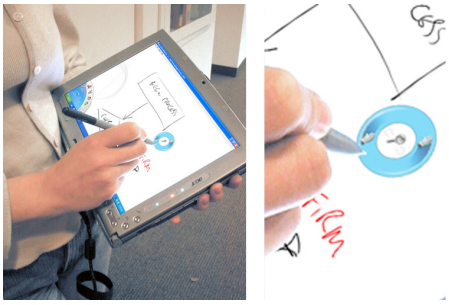


Figure 1. Pen-based environment and a tracking menu.

In this paper, we investigate a GUI-based solution that addresses the problems of rapid tool switching and tool palette round trips. This technique, called *tracking menus*, requires only pointing and pressing with the pen tip – no keyboard presses or physical buttons are required. Tracking menus can also be operated with a single-button mouse,

where pressing the mouse button simulates pressing the pen-tip.

Note that this work can be viewed from an alternative perspective. Consider a typical tool in a GUI as a single function being attached to the cursor (for example, the pan-hand icon replaces the cursor when panning is activated). This singularity of assignment is a fundamental convention that has remained unchanged for years in GUI design. Tracking menus evolve this convention. In effect, tracking menus function as a multi-headed tool – a way of having two or more tools simultaneously attached to the cursor. Our tracking menus implementation and design work explore this design space of multi-function tools.

In the remainder of the paper, we first describe the mechanics of how tracking menus operate. We then focus on a particular application – integrating a pan-and-zoom tracking menu in a commercial drawing program, SketchBook. Next, we report on a usability study of our design and discuss usability issues and user feedback we received. Lastly, we illustrate and discuss some design issues and variations of tracking menus and provide examples of other applications.

TRACKING MENUS

A tracking menu is a graphical user interface widget that is controlled by either a pen or mouse (see Figure 1). It is invoked and dismissed in the same manner as a traditional modal tool by clicking on a tool palette or menu item. Like traditional menus, a tracking menu consists of a cluster of graphical buttons. The cursor can be moved within the menu to select and interact with items. However, unlike traditional menus, when the cursor crosses the exterior edge of the menu, the menu is moved to keep it under the cursor.

Figure 2 shows a simple physical analogy to our tracking menus design. Consider moving a jar lid with the tip of a pencil. This can be done in two ways. The first obvious way is that the pencil can be pressed down into the lid and the lid dragged. The second way, which we use in tracking menus, is by contacting the sides of the lid. This results in the ability to move the lid without pressing down. Note that the pencil can be moved within the lid as well; the lid will remain stationary if the sides are not contacted. Figure 3 shows schematically how our tracking menu system corresponds to the jar lid example.

The interior of the tracking menu can be divided into multiple regions that are assigned to different types of functionality. This functionality can be invoked by pressing in the region. Also, the regions can have irregular shapes (an important design characteristic discussed later) and invoke discrete actions (such as buttons) or continuous actions (such as sliders).

(a)

(b)

(c)

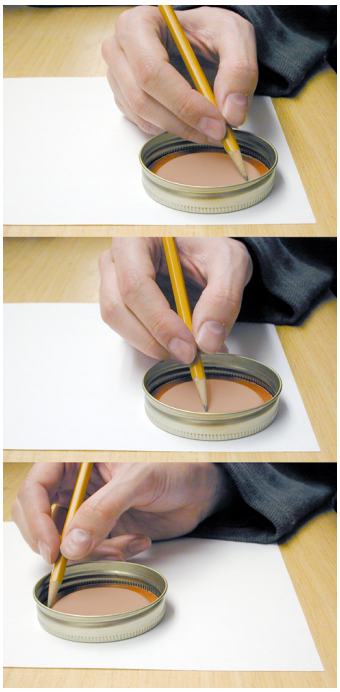


Figure 2. Physical analogy using a jar lid as the tracking menu and a pencil as the stylus. (a) initial state of pen and tracking menu (b) movement of pen without contacting edge – tracking menu remains stationary (c) pen contacts edge of lid – tracking menu moves with pen.

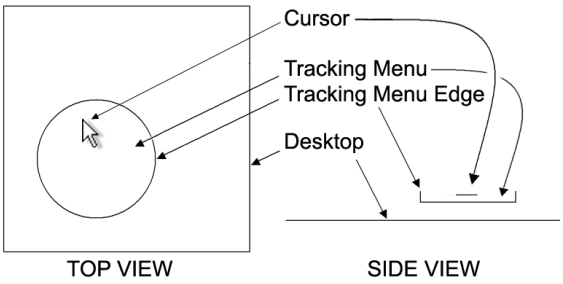


Figure 3. Tracking menu schematic design.

We implement tracking menus using the multiple input states sensed by pen computers (or using regular mouse events on standard keyboard and mouse configurations). Figure 4 shows the pen input states sensed by the Tablet PC. When the pen is more than approximately 1.5 cm above the tablet surface, it is *out-of-range* and the system does not track the location of the pen. When the pen is moved closer, the tablet begins *tracking* the tip of the pen and the cursor follows the tip of the pen. Finally, *touching* occurs when the pen contacts the tablet surface.

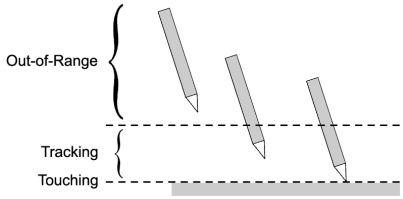


Figure 4. Input states of active digitizers.

Figure 5 shows a state transition diagram [4] of the tracking menu once invoked. When the pen is out of range (State 0) the tracking menu is visible but stationary. When the pen comes into range, the system ensures that the tracking menu is under the cursor, repositioning the tracking menu if needed. In this new state (State 1) the cursor is allowed to move freely within the tracking menu so the user can highlight menu items. Contacting the edge of the tracking menu transitions to State 1E and causes the tracking menu to move with the cursor. Alternatively, touching the pen down over a menu item invokes the item's action and transitions to touching (State 2). While in the touching state the tracking menu is typically hidden.

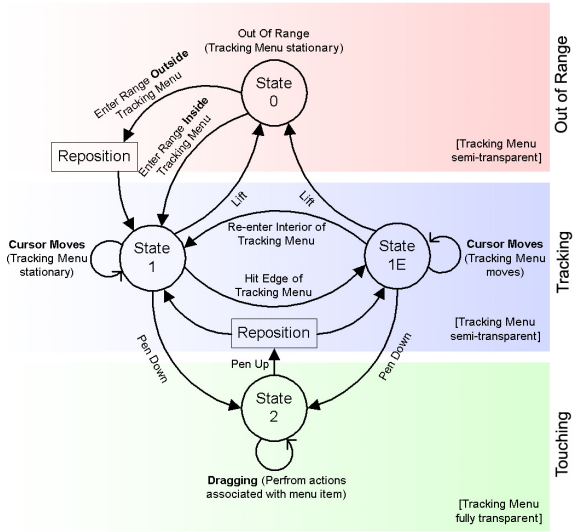


Figure 5. State Transitions of a tracking menu.

When using a mouse, the same set of state transitions apply except that Out Of Range (State 0) does not occur. State 0 simply adds the functionality of directly jumping to a new screen position. However, this result can also be achieved by moving the tracking menu in State 1E. Thus, tracking menus work both with the pen and mouse.

RELATED WORK

Tracking menus are related to a number of pop-up menu techniques such as: linear menus, pie-menus [6], floating pie-menus [17], control menus [16], flow menus [9] and marking menus [13]. All of these techniques, like standard graphical user interface techniques such as floating windows and tool palettes, allow users to keep subsets of commands close at hand. As named, this class of interaction controls all pop-up their menus at a fixed location and remain stationary until they are dismissed. In addition, flow menus and control menus integrate menu item selection with dragging, which is also a valuable design characteristic of tracking menus.

Techniques that employ mobile tool palettes are also related to tracking menus. Two-handed research input has proposed techniques for manually moving tool palettes closer to the work area or cursor (e.g., Toolglass [3] and T3 [14]). These mobile tool palettes follow the movement of a secondary input device and the user can select a tool using the primary input device.

Other techniques have been developed which do not require the user to manually position the tool palette. Baudel et. al. [2] proposed a tool palette that remains hidden until a button is hit on a non-dominant hand input device. This results in the palette being popped-up at a fixed position nearby the cursor. In effect the tool palette follows the cursor, but it must be explicitly brought up when needed and must be explicitly dismissed.

Tracking menus build on these previous systems by supporting similar functionality but are designed to operate entirely from a single input device providing one-handed input. Also, the tracking menu is persistent and does not explicitly need to be invoked or dismissed, further optimizing the interaction.

CASE STUDY

We now present a concrete design using tracking menus to solve a real problem in a commercial application. A set of usability tests was conducted to refine the designs. This case study highlights some of the interesting characteristics and features of tracking menus. Following the Case Study, we discuss the design space of tracking menus, including design characteristics, variations, and further design examples.

The target commercial application, Alias SketchBook, is a freeform sketching and painting tool designed specifically for pen-based Tablet PCs.[1] We have observed in our initial usage of our drawing program that zoom and pan tools are commonly used in rapid succession. For example, a user zooms-in and then adjusts the image by panning it over (or vice versa). The problem that we focused on was to reduce the transaction cost of switching between these tasks. As keyboard accelerators are not available on the Tablet PC, tool palette round trips would be needed to switch tools.

[1] A free download trial version of this program is available to demonstrate tracking menus www.dgp.toronto.edu/~akhan/sketchbook.

Tool palette round trips are exacerbated when combined with panning and zooming operations. These operations also require cursor movement — panning requires "pulling" the canvas from one spot to another and zooming requires a specific point to be indicated on the canvas as the point to be "zoomed in on". Dragging can then be used to indicate the amount of zoom. These operational movements tend to displace the cursor away from the palette thus increasing the length of the round trips and the time taken to perform them.

(a) (b)

Figure 6. Pan-and-Zoom tracking menu design. (a) Cursor hovers over pan region. (b) Cursor hovers over zoom region.

Figure 6 shows the tracking menu we have developed to address this problem. It consists of two main semi-transparent visual regions, to give it the properties of a see-through widget [10, 12]. Panning is placed in the outer ring and zooming on the inner region.

Tracking menus are designed to support an important user behavior we have observed called *pawing*[2]. An example of pawing is panning over a large image where several panning drags are required to reach the desired view. The user must repeatedly move the input device and then drag to move a significant distance. Typically these drags are performed ballistically, in quick repetition, and in a variety of directions. Traditional modal tools support this behavior.

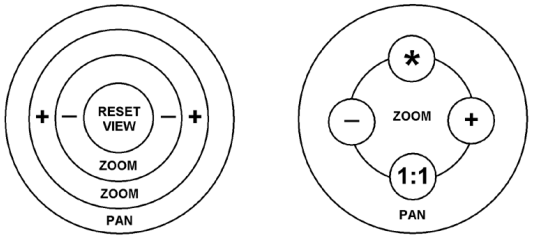
Ideally, the user should be able to make repeated dragging motions without accidentally changing the tool selected by the tracking menu. We achieve this behavior through a vitally important characteristic of the outer region of the tracking menu (for example, the pan region in Figure 6). It is very easy to select the outer region – all that is needed is a movement large enough to "hit" the edge of the tracking menu. The cursor cannot miss the edge because movement in any direction eventually results in hitting the edge. Furthermore, the edge region cannot be overshot since the tracking menu will move when the cursor contacts the edge. In this way, selection of the outer region is guaranteed by casually moving the tracking menu and pressing down. In effect, the user does not have to consciously pick the pan region on each drag. The net result is that pawing behavior is supported.

Based on our experience and observation of sketching tasks, panning is a more frequently used operation than zooming and therefore was a good candidate for the outer region. We placed the zoom operation at the center of the tracking menu because it is a less frequently used operation.

Because there is sufficient visual feedback (the canvas pans or zooms as the pen is dragged) during both of these operations, the tracking menu disappears once the user "pens-down" over a region and it reappears on "pen-up".

We have experimented with many additions beyond this basic design. Typically, drawing programs have many functions that are closely related to zooming and panning such as "reset view", "actual size", discrete and absolute zoom steps or settings (+/- 10%, 25%, 50%, 75%, etc.). Figure 7(b) illustrates our integration of some of these functions.

We found there were two basic approaches to laying out additional functions in the tracking menu. One approach is to add more concentric rings, effectively creating a bull's-eye visual style [15]. The other approach is to add buttons on top of the zoom and pan regions. Figure 7 shows examples of the two approaches. In practice, we found the button approach more appealing. Ultimately, the concentric ring approach has several disadvantages. Essentially all commands are the same shape, unlike buttons, where shape and location can be used to help identify and remember the location of commands. This ring symmetry also makes it confusing to users since the direction that the cursor must be moved is arbitrary. Finally, the inner rings do not have the "easy to select" property of the outer ring and therefore offer no advantage.



(a) (b)

Figure 7. Alternate Layouts. (a) Concentric ring layout. (b) Buttons overlaid layout.

While buttons are a viable approach for adding secondary functions to a tracking menu, the menu can become cluttered as the number of buttons increases. The solution we have used to combat this problem is to embed a pop-up

[2] Note that pawing is different from clutching. Clutching involves repositioning an input device without repositioning the cursor (a state 1 to 0 to 1 transition). However, pawing can be performed without requiring clutching (a state 1 to 2 to 1 transition).

menu in a button thus providing a location for extra functionality without further cluttering the tracking menu. More tertiary, less common, functions are placed in these menus. However, utilizing a marking menu [13] with these buttons still allows for very fast access. See Figure 8.

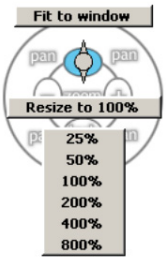
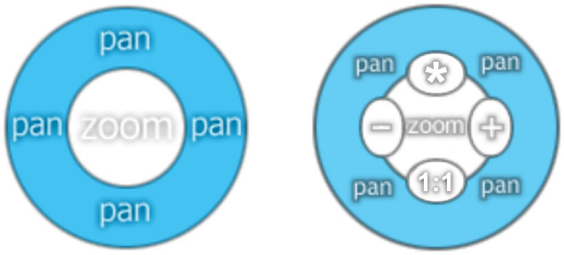


Figure 8. Tracking menu with marking menu activated from an embedded button.

Usability Tests

We conducted three rounds of usability tests: informal tests within our research group, formal tests with external users on some design variations, and formal tests with external users on the refined production design.

Within our own research group we refined our tracking menu design by informally testing the interaction on our group of four user interface researchers. From this work we derived two candidate designs.



(a) (b)

Figure 9. Pan-and-Zoom tracking menus (a) Basic Version and (b) Deluxe Version containing "+" to zoom-in (double the current zoom factor), "-" to zoom-out (half the current zoom factor), "1:1" to return to the original non-zoomed canvas scale, and "*" to launch the marking menu.

The first design was a very simple tracking menu with only pan and zoom rings (see Figure 9(a)). The second design was the deluxe version shown in Figure 9(b), which included the basic design plus interior buttons, including a button to launch a marking menu.

Using these designs, we observed that there are some conditions where precise positioning of the tracking menu is cumbersome. If the user overshoots a small target, there is the added step of having to move across the tracking menu to the opposite border to fine-tune its position. However in the following user tests this issue was not reported, as the task did not require precise positioning.

In the first round of testing with external users, six people were tested. They were artists with drawing skills (experience with Adobe Photoshop was typical) and computer literate managers. Only the artists had experience with pen-based systems (typically Wacom tablets on desktop systems). Users were given eight tasks which required the use of panning and zooming functions. In each task, the user was presented with a screen image of a *before* picture followed by an *after* picture on a piece of paper, which was a portion of the picture with some zooming and panning operations applied (see Figure 10). They were then asked to make the screen image look like the *after* picture using the functions on the tracking menu. Users were given no explanation on how the tracking menus worked. They tried both the simple tracking menu and the advanced tracking menu and were asked to think-out-loud. The tester in the room recorded their comments and his own observations.

(a) (b)

Figure 10. (a) Original "before" image. (b) Target "after" image for user study.

After a few minutes of learning while performing the tasks, all users successfully completed the task set. The interesting observations we made include:

- People new to a pen interface typically expected the relative movement of a mouse.
- While users did not need the commands on the buttons in the tracking menu to perform the task (only pan and zoom were needed), users saw these buttons and were unsure what the button labels meant. Some users expressed that they thought these buttons were arithmetic functions while others ascribed no meaning to them. We believe this could be fixed with better labels or tool tips.
- Several users tried to work on the right-hand side of the screen to avoid obscuring their target with their hand, while others clicked directly on the target and zoomed, using the tracking menu as designed.

- As expected, users easily found and used the larger zooming and panning regions of the tracking menu and were not distracted by the smaller buttons.

- Our design to support pawing seemed to be effective. We observed users casually panning, not fully aware that they had to stay in the panning ring, but they panned successfully nonetheless.

In our second formal user study, we implemented the pan-and-zoom tool within the SketchBook application and tested it in context with the other drawing and manipulation tools (see Figure 11). A separate usability team for SketchBook conducted this study. Their objective was to determine if the technique was mature enough to be released in a product. To keep things simple we limited our testing to the basic design shown in Figure 9(a) hoping to pursue the deluxe design later if the basic design passed testing.

Six art college students with varying degrees of computer exposure were invited. As in our first study, participants were tested without training or guidance to perform the same navigation tasks. Participants were asked to think-out-loud and the tester in the room recorded their comments and observations. A group of four usability designers also observed via a video camera and made notes.

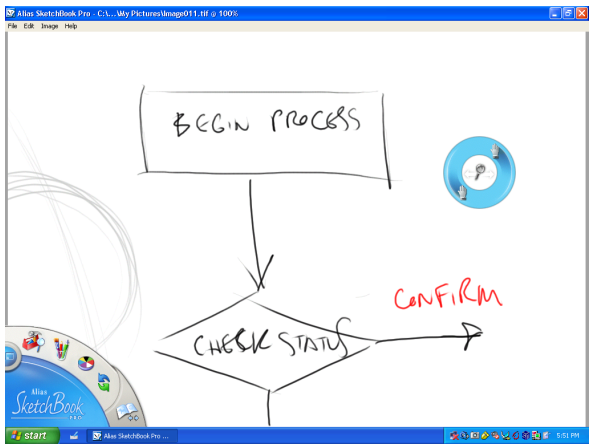


Figure 11. SketchBook application with pan-and-zoom tracking menu active.

In this case, participants performed the set of tasks in two rounds. For one of the rounds they performed the tasks using a separate zoom tool and pan tool. The other round involved using our combined pan-and-zoom tracking menu to perform the task. Half of the participants used the separate tools for the first round, and then used the combined pan-and-zoom tool for the second round. Ordering was reversed for the remaining participants to counterbalance ordering effects.

From this study, the following observations were made.

- In the separate pan and zoom tools, all participants stated that they did not like having to go back and forth to the tool palette to switch tools.

- All participants had a learning curve to figure out the combined pan-and-zoom tool and were mostly comfortable with it within a few minutes.

- Two thirds of the participants preferred the combined pan-and-zoom tool over the separate tools.

- Two thirds of the participants felt that they completed the tests faster with the combined pan-and-zoom tool. Our informal timings of their task completion times matched or exceeded users' perceptions. In fact, one participant who subjectively did not favor the pan-and-zoom tool still had better task completion performance using the pan-and-zoom tool.

One participant stated, "It's actually easier to use this [pan-and-zoom tool], once you've figured out that it follows you." Another participant found the combined pan-and-zoom tool very enjoyable: "I kind of like this – I could sit around and do this for hours."

Not all participants enjoyed using the combined pan-and-zoom tool. For example, one participant stated: "This [pan-and-zoom tool] seems like a very large tool for the job." Here, the participant was referring to the visual size of the tracking menu. A few participants were quite confused over the graphic icons for the zoom region in the combined pan-and-zoom tool. They initially thought the icons were buttons to perform one-shot zoom operations.

All of our usability studies showed that the combined pan-and-zoom tool was effective and worthy of incorporating into our commercial sketching program. While we would have liked to pursue the "deluxe" version of the pan-and-zoom tool, the compressed schedule for delivering version 1.0 of SketchBook made this impractical. Therefore, the simple pan-and-zoom tool was implemented for the product.

Some additional design details were important to make the pan-and-zoom tool interact more seamlessly with the main tool palette of SketchBook located at the bottom of the application window (see Figure 12). First, conceptually, we wanted the pan-and-zoom tracking menu to slip under the main tool palette. Secondly, we wanted the user to be able to select new tools from the main tool palette without having a jarring visual transition when switching tools. We observed this jarring visual transition when the pan-and-zoom tool was prevented from continuously slipping under the main tool palette, and instead disappeared immediately (see Figure 12). To preserve visual continuity we added a dotted outline of the pan-and-zoom tool for those portions

of the tool within the main tool palette as an "x-ray" effect. This design achieves our goals and provides effective feedback and orientation for the user.

Figure 12. Pan-and-zoom tracking menu under tool palette with x-ray feedback.

Since our initial product launch, we have received positive feedback from our growing user community. While we are still in the early stages, initial reaction to the overall user interface design of the pan-and-zoom tool is positive.

Design Characteristics and Variations

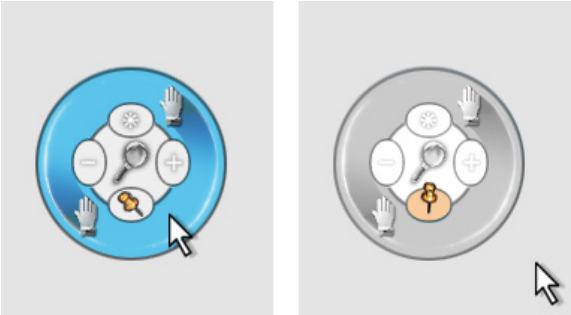
In this section we describe further enhancements that can be added through layout and boundary design, activation algorithms, and careful interaction with other interface elements. By doing so we illustrate the design space of tracking menus.

Pinning the Tracking Menu: Temporary Deactivation

Depending on the user's workflow, it may be desirable to separate the cursor from the tracking menu. For example, in our SketchBook application, users desire the ability to rapidly switch between the pan-and-zoom tool and drawing. To accommodate this type of feature, we have developed the notion of a pushpin and lock (see Figure 15). When the user selects the pushpin button, the tracking menu is temporarily deactivated, remaining posted and stationary; it grays out to indicate the inactive state. The cursor can now leave the tracking boundary edge. The next time the cursor travels into the tracking menu the pushpin is automatically released and the tracking menu behaves as normal (i.e., moving when the cursor hits the tracking border). The notion of a lock was developed to explicitly pin the tracking menu and not release it until the lock is explicitly selected again. Note this enables fluid transitions between panning, zooming, and drawing with a brush, as brought up in the user tests.

Dividing the Exterior Region

There are a variety of ways to divide the exterior region. Figure 13 shows how regions can be laid out so that some functions are easy to invoke by being placed against the edge of the tracking menu. This characteristic allows selection by direction of movement rather than only by position. This characteristic has been exploited in other GUI techniques (e.g., marking menus [13] or T-Cube [18]) and has characteristics similar to goal crossing tasks as proposed by Accot and Zhai [1].



(a) (b)

Figure 15. Tracking menu pushpin. (a) Pushpin about to be engaged. (b) Pushpin is now active and tracking menu grays-out; cursor can leave tracking menu border. Once cursor crosses back into tracking menu, pushpin automatically disengages.

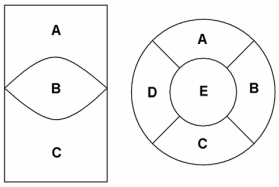


Figure 13. Layout of exterior regions of tracking menus.

Dragging Algorithms for Tracking Pen Movement

While moving the tracking menu in the tracking input state, a variety of dragging algorithms can be employed. We have used the simple physical approach which moves the tracking menu at the point of cursor contact with the tracking menu edge and keeps the cursor stuck at the edge until the user "backs up" a bit. Alternatively, we could use a different dragging algorithm such that the cursor gets attached to the tracking menu edge but can go beyond the edge and drags the tracking menu through a metaphorical string or elastic. Simulating gravity and weight for the tracking menu and imparting forces through cursor activity is possible and may add a fun factor to the technique.

Tracking Boundaries

The visual boundary of the graphical representation of the tracking menu does not have to map directly to the tracking boundary (see Figure 14). The tracking boundary can have a different shape and it can be larger or smaller than the visual boundary. Moreover, tracking boundaries can be non-contiguous. For example, there could be a hole in the tracking menu or interior tracking menu boundaries (*walls*), can be defined. Interior walls may be useful to bias the space and allow the cursor to remain in a sub-region more easily.

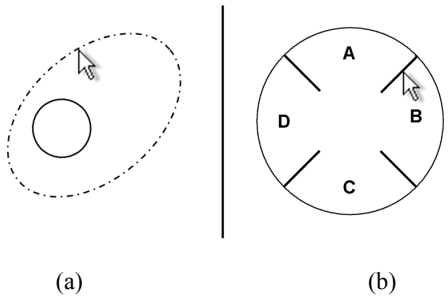


Figure 14. (a) Tracking and graphical borders need not match. Here we see a circular graphical tracking menu with an elliptical border. (b) Shows interior tracking boundaries.

Design Examples

We now present a few examples to illustrate different applications and design possibilities for tracking menus. Note that we did not build interactive prototypes of these designs and the designs themselves may be incomplete. However, they are intended to show possible design variations and to stimulate thought.

User-Assignable Functions for the Outer Region

Figure 16 shows a tracking menu used to implement some common functions of a drawing application tool palette. In this design, selecting a function assigns the function to the outside region of the tracking menu. Thus, subsequent button presses and drags in the outside region engage the assigned function. Another design variation is a split region technique shown in Figure 16(e-f) where two functions can be assigned to different sections of the outside region. Thus two frequently used tools can be selected by quick ballistic movements to the left or right followed by a button press.

Numeric Entry

Figure 17 shows a tracking menu configured to act as a numerical keypad. The hole in the center of the keypad allows the tracking menu to be aimed at different numeric fields. Pressing on the various keys enters numbers as expected. The hole has the special property that it allows the field to be edited as expected — the text cursor can be positioned in the text field by clicking between numbers, and numbers can be selected by dragging, etc. Furthermore, the tracking menu can snap to the position of the numeric field thus making it easier to aim the tracking menu.

3D Camera Control and Permeable Zones

Figure 18 shows a tracking menu for controlling the position and orientation of a viewpoint in a 3D scene. Typically, this is called a 3D virtual camera and involves several separate tools for panning, zooming, and tumbling (orbiting the camera about the center of the 3D scene). Furthermore, there are other types of camera movements that can be used such as roll, yaw, and pitch. These controls

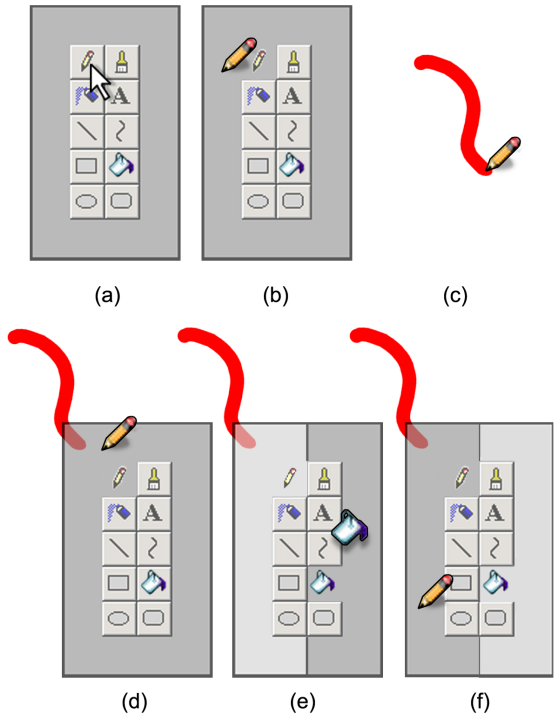


Figure 16. Tool palette tracking menu. (a) Initial state. User selects pen tool. (b) Pen tool assigned to exterior region of tracking menu. Cursor changes to pen icon. (c) User pens-down and tracking menu becomes invisible during the drag operation, making a red mark. (d) Tracking menu reappears on pen-up event, repositioned under cursor. (e) User selects second tool; the flood fill tool. Exterior region of tracking menu is divided into two regions. (f) When cursor moves to the left side of the tracking menu, the pencil tool is enabled and cursor changes to pen icon.

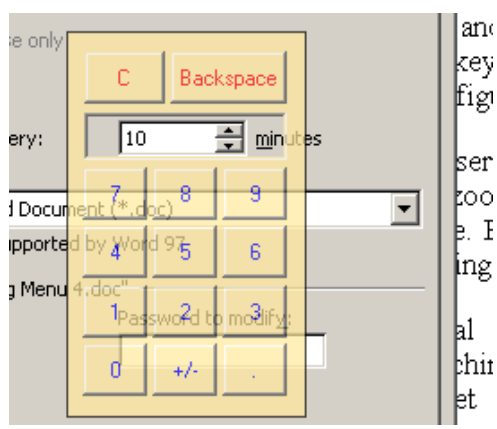


Figure 17. Numeric pad tracking menu.

form a nice cluster of functionality that can be made available via a tracking menu. Figure 18 shows a tracking menu where the most frequently used camera control (tumbling) is given priority in the design by being placed in the large outer region. Additional, less frequently used commands are placed appropriately in smaller regions thus reducing the chance of accidental engagement. Note that this design explores the usage of three *permeable* zones (reset view, undo and redo). Here the user must dwell over the region border with the cursor and after some time (approximately half a second) may enter and activate the zone. This provides a way of offering functionality within the tracking menu but at a reduced level of accessibility.

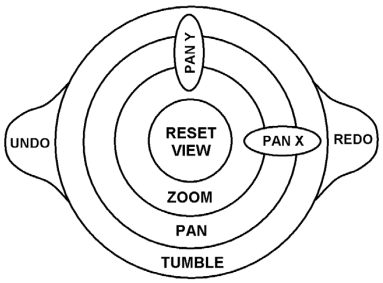


Figure 18. 3D camera control tracking menu.

CONCLUSIONS & FUTURE RESEARCH

In this paper we introduced a new GUI widget called the tracking menu designed to provide rapid switching between different modes or actions using only a pen tip and no external signals.

In the future we wish to explore the issue of display size. We believe this technique will be particularly effective in large wall-sized displays [5, 7, 8] where navigation tasks and travel time problems are acute.

Our timings from our third usability study indicate tracking menus have a time performance advantage over tool palette round trips. Future research could measure this in a controlled experiment.

We believe this design can work well in a variety of hardware configurations beyond the targeted single pen-tip configuration. Our experience has shown that the technique works well using a standard mouse device. Moreover, tracking menus can be adapted to work in systems that employ two input streams (e.g., trackball and stylus), one for each hand.

While we have found this technique to be extremely useful for small sets of functionality, we are still investigating whether or not the tracking menu technique can scale to larger sets of hotkeys, or ultimately, even full keyboard replacement.

ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of the SketchBook team and the Usability team, in particular, Lynn Miller, Joe DiVittorio, Marsha Leverock, Ian Ameline, Alex Tessier, Ken Xu, Tom Wujec, and Mark Charlesworth. We greatly appreciate the help of Ken Hinckley in refining this paper.

REFERENCES

1. Accot, J. Zhai, S. (2002) More than dotting the i's – foundations for crossing-based interfaces. *Proceedings of ACM CHI 2002*, 73-80.
2. Baudel, T., Buxton, W., Fitzmaurice, G., Harrison, B., Kurtenbach, G. and Own, R. (1995) Clickaound tool-based graphical interface with two cursors. US Patent #5,666,499.
3. Bier, E. A., Stone, M. C., Fishkin, K., Buxton, W., Baudel, T., (1994) A Taxonomy of See-Through Tools. *Proceedings of the ACM CHI 1994*, 358-364.
4. Buxton, W. (1990). A Three-State Model of Graphical Input. In D. Diaper et al. (Eds), Human-Computer Interaction - INTERACT '90. Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 449-456.
5. Buxton, W., Fitzmaurice, G., Balakrishnan, R., and Kurtenbach, G. (200) Large Displays in Automotive Design. *IEEE Computer Graphics and Applications*, 20(4), pp. 68-75.
6. Callahan, J., Hopkins, D., Weiser, M. & Shneiderman, B. (1988) An empirical comparison of pie vs. linear menus. Proceedings of CHI '88, 95-100
7. Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, EG., Janssen Jr., W.C., Lee, D., McCall, K., Pedersen, E.R., Pier, K.A., Tang, J., Welch, and B. (1992) Liveboard: A Large Interactive Display Supporting Group Meetings, Presentations, and Remote Collaboration. *Proceedings of ACM CHI 1992*, 599-607.
8. Guimbretière, F., Stone, M. & Winograd, T. (2001) Fluid Interaction with High-resolution Wall-size Displays, *Proceedings of ACM UIST 2001*, 21-30.
9. Guimbretiere, F. & Winograd, T. (2000) FlowMenu: Combining Command, Text, and Data Entry. *Proceedings of ACM UIST 2000*, 213-216.
10. Harrison, B., Kurtenbach, G., Vicente, K. (1995) An Experiment Evaluation of Transparent User Interface Tools and Information Content. *Proceedings of ACM UIST 1995*, 81-90.
11. Harrison, B., Fishkin, K., Gujar, A., Mochon, C., Want, R. (1998) Squeeze me, hold me, tilt me! An exploration of manipulative user interfaces. *Proceedings of ACM CHI 1998*, 17-24.
12. Kramer, A. (1994) Translucent Patches: Dissolving Windows. *Proceedings of ACM UIST 1994*, 121-130.
13. Kurtenbach, G. & Buxton, W. (1993) The limits of expert performance using hierarchical marking menus. *Proceedings of the ACM CHI 1993*, 482-487.
14. Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B., (1997) The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency, *Proceedings of ACM CHI 1997,* 35-42.
15. Kurtenbach, G., (1993) The Design and Evaluation of Marking Menus, Ph.D. thesis, University of Toronto, Dept. of Computer Science.
16. Pook, S., Lecolinet, E., Vaysseix, G., and Barillot, E. (2000) Control Menus: Execution and Control in a Single Interactor. *Proceedings of ACM CHI 2000 Extended Abstracts*, 263-264.
17. Rubio, J.M. and Janecek, P. (2002) Floating Pie Menus: Enhancing the functionality of Contextual Tools. *Proceedings of ACM UIST 2002 Conference Companion*, 39-40.
18. Venolia, D. and Neiberg, F. (1994) T-Cube: A Fast, Self-Disclosing Pen-Based Alphabet. *Proceedings of ACM CHI 1994*, 265-270.

NOTES