EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2009)
C. Grimm and J. J. LaViola Jr. (Editors)

Shadow Buttons: Exposing WIMP Functionality While Preserving the Inking Surface in Sketch-Based Interfaces

Diane Marinkas[1], Robert C. Zeleznik[2], and Joseph J. LaViola Jr.[1]

[1]University of Central Florida, School of EECS, Orlando, FL USA†
[2]Brown University, Department of Computer Science, Providence, RI USA‡

Abstract

We present Shadow Buttons, an approach to placing WIMP interface elements into gestural and sketch-based interfaces. Utilizing the hover state, supported by pen-based devices such as Tablet PCs, we provide users with important WIMP-based functionality by invoking widgets only when the stylus hovers over "shadow" regions. In this way, every pixel of the display, including the shadows, can be drawn on. By interacting with a shadow region while in the hover state, users can temporarily display and interact with familiar WIMP elements. We explore the Shadow Button design space as it relates to handwritten mathematical expressions and also present an informal evaluation of our technique, examining various Shadow Button parameters including button size, placement, and method of invocation. Preliminary results indicate that users prefer utilizing the hover state over tapping on the Shadow Button and in general, prefer interface elements to be as close to the ink as possible. In addition, the distance between the shadow region and the menu was found to be the most important factor in Shadow Button usability.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Computer Graphics]: User Interfaces—Interaction Styles

1. Introduction

Pen-based interfaces are a simple and natural method of interacting with computers. For example, digital artists frequently use Wacom tablets for intuitive, free-form input, and students often choose tablet PCs to ease note taking in class. Handheld devices such as personal digital assistants (PDAs) are adopting pen-based interaction, as are cell phones, ultra-mobile PCs (UMPCs), and mobile internet devices (MIDs). As these devices become smaller and more feature-rich, we are faced with the problem of shrinking screen space availability.

One method of counteracting this problem is to use a gestural interface, where the user performs a particular movement to initiate a command. Since there are few or no WIMP (Windows, Icons, Menus, and Pointing devices) interface elements, the majority of the screen can be devoted to the stylus input area. Additionally, a gestural interface can be "mode-less", where a series of operations can be performed without an explicit mode switch from one operation to another. However, there are inherent difficulties with this interaction technique. For instance, gestural interfaces suffer from being non-self-disclosing. In most cases, a user cannot simply sit down and start using the application without human assistance or training. Another issue is the increased cognitive load placed on the user. The user must be able to remember all of the gestures, which artificially creates an upper bound on the number of gestures and consequently, the complexity of operations which can be performed with the software [BZW*09].

We believe that Shadow Buttons will alleviate many of these issues. A Shadow Button is not a button in the true sense, but rather a shaded region found within or next to the bounding box of a particular collection of digital ink strokes (i.e., a word, a phrase, or, as in our case, a mathematical expression). Shadow buttons make use of the "hover" state employed by tablet PCs in which the tablet can detect the loca-

† {marinkas,jjl}@eecs.ucf.edu
‡ bcz@cs.brown.edu

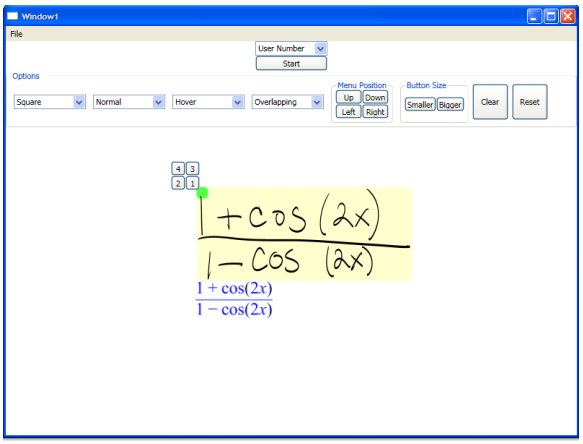

Figure 1: *The application used for our evaluation.*

tion of the stylus when it is a short distance above the tablet surface. Using the stylus, the user hovers over the shadow region invoking a floating widget containing a menu of "real" buttons. These buttons correspond to functionality that could be found in a drop-down menu in a traditional application.

Using our technique, we can include WIMP elements in a pen-based application without consuming screen real estate. All of the complex functionality of the application can be accessed from the Shadow Button's menu, freeing up the inking space for additional input. This technique has the added benefit of being easily discoverable by the user.

In this paper, we present an analysis of the Shadow Button design space by examining various interface parameters including button size, button placement, menu shape and location, and invocation method. In addition, we discuss the results of an informal usability study exploring how these different interface parameters effect users' reactions to the technique. Specifically, we were interested in finding whether utilizing the hover state or a tap on the Shadow Button was preferable. In the next section, we discuss work related to Shadow Buttons, followed by a discussion of our design criteria and the Shadow Button design space. Section 4 discusses our informal evaluation followed by our results in Section 5. Section 6 presents areas for future work and concludes the paper.

2. Related Work

There have been several in-band menuing strategies for invoking application functionality without the explicit use of traditional WIMP-style interfaces. Marking menus provide users with the ability to invoke menu items by placing the menu under the pen-tip. All menu items are arranged the same distance from a fixed point, essentially on the circumference of an imaginary circle [KB94, CHWS88]. For this reason they are often called "radial" or "pie" menus. Users can drag the pen to the desired menu item or perform a "flick" gesture to select the desired menu option. Marking menus are advantageous because each choice is the same distance from the point where the user invoked the menu. Additionally, the target selections can be a bit larger than items in a linear menu, which adds to their ease of use.

Flow menus [GW00] are similar to marking menus, but were designed with pen applications in mind. The flow menu is also arranged radially into octants. Users employ a pen input device and, once the menu is invoked, move the pen from the center "rest" area to the octant they wish to select, and back to the center rest area. Users also have the option, depending on the selection, to seamlessly continue their interaction without lifting the pen. For instance, if a user selects a "move" operation from the menu, he may continue to drag-move the object without an explicit mode switch. Other similar menu techniques have also been developed such as tracking menus [FKP*03]. In these systems, users must tap on the screen to instantiate the menu. The act of tapping on the screen prevents users from entering ink and interrupts work flow. With Shadow Buttons, users can invoke a menu while the pen tip is hovering over the screen and still enter ink in the shadow region, saving screen real estate for inking.

Grossman's Handle Flags [GBH09] were designed to assist users with selection tasks in pen-based applications. Handle Flags are floating widgets that strive to provide a handle for accessing each potential selection in an inking application. A common task in inking applications is selection of the digital ink itself, rather than selection of a button or menu item. This is a difficult task, however, due to the fact that words or expressions can be composed of arbitrary strokes which may often overlap. With Handle Flags, when the user's pen approaches a potential selection, the Handle Flag fades in at a certain offset from the pen location. The user can then tap on the flag corresponding to the desired selection. Although Handle Flags are similar in spirit to Shadow Buttons, they are focused on selecting ink rather than invoking various interface widgets.

Closely related to our work is Hover Widgets by Grossman et al [GHB*06]. In this work, the tablet PC hover state is also used. When the user wishes to invoke the widget, he or she performs a simple L-shaped gesture very close to but not touching the tablet surface, then tapping the pen to the surface. The widget, which in this case is a single floating button, appears very near the user's cursor. The gesture "tunnel" is in the shape of an L in order to prevent accidental activation, since variations on the gesture's orientation (i.e., upside down) are unlikely to be intentional. Hover Widgets are the first effort recognized in the literature to make use of the hover state.

Shadow Buttons seek to extend the notion of hover widgets by once again employing the hover state while simplifying the method of invocation. Instead of invoking a widget using a gesture and tap combination, Shadow Button users

can simply move the stylus over the shadow region to invoke the menu. Additionally, a small menu of WIMP buttons immediately appears, rather than the additional step of tapping a single button to invoke a circular menu as with hover widgets.

3. Shadow Buttons

3.1. Design Criteria

Shadow Buttons are not buttons in the traditional sense. Instead, they provide shadow regions on the drawing or writing area where users can still use the pen to draw ink, but use the hover state to instantiate WIMP interface elements. In creating Shadow Buttons, we considered the following:

- **Screen space** - Our primary goal is the preservation of screen space. By invoking the buttons only when needed and with hover, we don't waste space for menus and toolbars. Additionally, every pixel of the display is available for inking, including the shadow regions.
- **Discoverability** - Shadow Buttons, unlike many elements in pen and gestural interfaces, are easily discoverable. The user may accidentally hover over the shadow region and make the widget appear, or he or she may see the differently colored region and be curious about its significance.
- **Convenience** - We believe that application functionality should be easy to access at a moment's notice. The shadow region is associated with a particular group of ink strokes and contains only commands that pertain to that grouping. For instance, in our work we have focused on mathematical expressions. A matrix of real numbers may invoke a widget containing matrix operations, such as finding eigenvalues or computing the inverse [ZMLL08].
- **Work flow** - Specifying commands to the application should not disrupt the user's work flow. Searching for an instruction in a series of traditional WIMP menus takes too much time and distracts the user from the task they want to accomplish. Keeping the necessary commands local means the interface stays transparent to the user.
- **Cognitive load** - Rather than have the user remember numerous gestures, we keep the frequently needed options in a menu, making it a "cheat sheet" of sorts. Additionally, the buttons and menus are familiar to most users and further reduce the amount of "thinking overhead" needed to use the application.

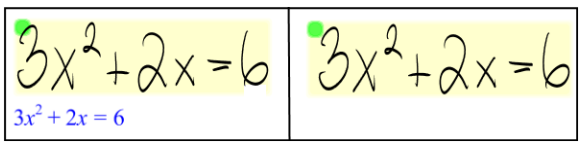

Figure 2: *An example of the shadow region overlapping the ink (left) and not overlapping (right).*

3.2. Design Space

In realizing the Shadow Button concept, we explored a rather large design space. In this paper, we decided to focus on Shadow Buttons in the context of mathematical sketching systems [LZ04]. Thus, we examined them as they apply to handwritten and recognized mathematical expressions. In particular, we have implemented Shadow Buttons in MathPaper [ZMLL08], a pen-based system for fluid entry and editing of mathematics with support for interactive computation. Users use the stylus to input handwritten mathematics, and the application recognizes expressions and can solve and graph equations, integrals, matrices, etc. The user's digital ink appears surrounded by a light yellow bounding box. The shadow region is a small green square in the upper left-hand corner of this bounding box (see Figure 2). When the user hovers over this region, four small, square buttons appear in a square configuration. Three of the buttons perform frequently used operations such as "Simplify Expression," "Graph Expression, " and "Drag Interactively." When the user selects the fourth button, a traditional drop-down menu appears presenting the user with more options.

There are many design parameters which may be tweaked to change the user experience. There are two categories of parameters that we explored: those that have to do with the shadow region, and those that have to do with the menu widget that is presented to the user. For instance, the location of the shadow region belongs to the first category, whereas the size of the buttons belongs to the second. Here we discuss only a subset of the possible configurations.

The parameters related to the shadow region include location and size. When we refer to the location of the shadow region, we simply mean in which corner of an expression's bounding box it is in. We chose to place it in the upper left-hand corner, but another good choice would have been the lower right-hand corner. In either case, the objective is to be unobtrusive. We do not want to interrupt the work flow of the user by accidentally triggering the widget and having it become a nuisance to the user. We feel it is the most out of the way in the upper-left corner, since we assume the user will be writing in a left-to-right direction. For the same assumption, the lower-right corner also makes sense since the user will finish writing and have several common commands at their disposal without any additional hand movement.

Another aspect of the location variable is its distance relative to the digital ink. In our implementation, it is fixed by the corner of the ink bounding box (see the left image in Figure 2). This means that the ink is displayed overlapping the shadow region. Another prototype expanded the bounding box and shifted the shadow region to the left, so that the region does not interfere with the ink, as shown in the right image in Figure 2. We believe that some users may be distracted by the overlap with the ink, or that it may cause accidental triggers. In the future, we would like to expand on this idea and have an option where the user may circle the

shadow region and drag it out of the way to a comfortable distance.

The size of the shadow region is also of particular interest. The region must be large enough to be useful, but small enough to prevent accidental invocation. In our first prototype, the region was approximately 25 pixels by 25 pixels, and we experienced a moderate number of unintentional triggers. We count accidental triggers as those that both invoke the widget when not specifically called upon and "catch" on one of the buttons, preventing the collection of ink. A second implementation used a much smaller region (9 x 9 pixels) which helped reduce the undesired triggers. Another option is to make the size of the shadow region proportional to the size of the bounding box surrounding the ink.

In addition to the shadow region, the menu itself has many variations. We assumed that a square arrangement of the menu buttons was optimal, but we also experimented with others. We tried aligning the buttons vertically, going both up from the shadow region as well as down. Additionally, we aligned the buttons horizontally. We believe that a circular arrangement of buttons, perhaps combined with a marking menu, may be a desirable alternative, though this remains untested.

Another parameter of interest is the way in which the widget is invoked. Shadow buttons were created with tablet PCs and Wacom tablets in mind. These devices have a hover state, where the stylus position can still be detected within a small distance above the surface of the tablet. We chose to use the hover state rather than a gestural command in an effort to reduce accidental triggering of the widget. However, a different approach, such as tapping to bring up the menu, would be more appropriate for a handheld device which may not report hover data.

We also believe that button size plays a role in the utility of Shadow Buttons. Depending on the arrangement of the menu buttons, the size of the buttons may increase or decrease the number of accidental invocations. Another problem is that if the buttons are too large, they may obscure the ink or other elements of the UI.

The final parameter which we considered straddles both categories and has the most nuances. The distance between the shadow region and the location where the widget appears is very important. The menu must appear close enough to the shadow region to be accessed quickly, without interfering with other elements on the screen. It must also be far enough away to not be triggered accidentally. Additionally, depending on the menu shape, the position relative to the expression may change. By default, the square menu appears directly diagonal to the shadow region (see Figure 1).

4. Evaluation

We performed an informal pilot study to determine if user preferences coincided with what we felt were the important attributes of Shadow Buttons. In addition, we wanted to gauge users' reactions about different Shadow Button configurations. This study consisted of nine participants, all members of the Interactive Systems and User Experience Lab at the University of Central Florida. This had the advantage that most were experienced tablet PC users and therefore had an idea of the necessary elements in a pen-based application. However, this also a hindrance, since outside input from novice users would have allowed us to make more general conclusions. Seven participants were male and two were female, and two of the participants were left-handed.

We used a stripped-down version of MathPaper for the experiment (see Figure 1). We did not want to distract users with the extensive features of the full application. The program was implemented using Windows Presentation Foundation (WPF) and run on a HP tc4400 tablet PC. The program contained the necessary elements, such as a digital ink collection canvas, a math recognition backend, and bounding boxes for recognized mathematics. The software also had a large ribbon area consisting of several combo boxes and buttons in order for the tester and the users to change the Shadow Button configuration on the fly. Each time an option was changed, the current configuration was written to a log file for further analysis.

Each participant was asked to write several mathematical expressions. For each expression, they were asked to select a different button from the Shadow Button menu configured in several different ways. When selected, the button played a familiar Windows system sound. In the first test, the participants wrote a mathematical expression and selected a button from the menu in the "square", "up", "down", and "horizontal" configurations (see Figure 3). The second test consisted of writing a different expression and tapping or hovering over the shadow region to bring up the menu. The third example tested whether users preferred the shadow region to overlap the ink or not. After each test, the participants were asked which configuration they preferred and why.

The final test was "user's choice." Each user was asked to write an expression once on the left side of the writing area, once in the middle, and once on the right edge, to determine if Shadow Button configurations depend on screen location. In each case, they were asked to modify any of the Shadow Button parameters in the way they felt was ideal for all three cases.

5. Results

Of the nine participants, five preferred the square configuration of the menu buttons, three preferred the down configuration, and one liked the up configuration, though there was no consensus on a reason. Seven of the participants preferred hovering over the shadow region to tapping on it to invoke the widget. Two preferred tapping, and in fact, tapped even when hovering was enabled. When asked to tap on the

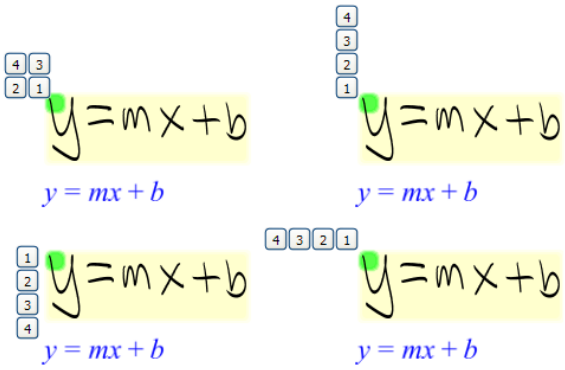

Figure 3: *The four menu shapes used in our evaluation. Clockwise from top: square, up, horizontal, down.*

shadow region, most users understood how to invoke the menu but did not understand how to get rid of the menu. One user attempted to select a button from the menu without lifting the stylus, as would be the case when using a flow menu. Seven users preferred the non-overlapping Shadow Button, one thought overlapping was better, and one said they were about the same.

Even though there was no consensus on menu shape, the participants' comments were similar. Those that preferred the square menu liked it because it kept the functionality within easy access. Each button was about the same distance from the shadow region. Those that liked the down configuration preferred it because it ran along the left edge of the expression's bounding box. In contrast to the up configuration, this kept the buttons close to the ink. Additionally, if the buttons appeared in a specific order, for instance, in order of most important to least important, the down shape lends itself nicely to this. Two of the participants said that they would have liked the horizontal shape if the buttons had run along the top edge of the bounding box, rather than heading off to the left as was the default horizontal configuration.

During the user's choice test, five users did not change the button size, three increased the size, and one attempted to decrease the size (they couldn't however because the button content would no longer be visible). Three again selected the "down" menu shape. All participants chose to move the menu from the default location, regardless of configuration. Only one participant chose to move the menu farther away from the expression, so it appears that in general users want the buttons as close to the ink as possible without obscuring it.

Preliminary results from our study indicate that the distance from the buttons to the shadow region is the most important factor to the usability of Shadow Buttons. Menu shape also plays an important role, but this reduces to a question of distance as well. In addition, the fact that the majority of our participants preferred using hover over tapping to invoke the buttons indicates that the Shadow Button concept is one users are willing to accept.

6. Conclusions and Future Work

Though our informal study was brief and entirely qualitative, we can identify specific areas for further exploration. We would like to run a formative usability study and collect quantitative data about Shadow Buttons in an ecologically valid setting. We would like to have participants attempt to complete a task both with and without Shadow Buttons. For instance, the objective of the task could be to hand-write a mathematical equation, recognize it, solve it, and graph it. Once the math is written, the other tasks may be present in the Shadow Button menu.

Additionally, we would like to further explore which menu shape is preferable, taking into consideration what we have learned in this pilot, and collect quantitative data. We would like to include a horizontal menu that by default runs along the top of the expression's bounding box. It would also be beneficial to collect timing data and distance information. Since our pilot study indicated that distance was the most important factor in determining the utility of Shadow Buttons, we would like to see to see how far away we can move the buttons before utility declines. We also want to explore what the shadow should look like. For example, is the green shaded region too distracting? Would a "real" drop-shadow be too subtle?

Finally, we recognize this work was tailored toward handwritten mathematical expressions. We would like to explore how Shadow Buttons might work in the context of other pen- and sketch-based applications and how the design space might change accordingly. For example, would it be plausible to use Shadow Buttons in an application that recognized words instead of mathematical expressions? Or what about in an application that focuses on hand-drawn sketches?

Since our participants were experienced tablet users, many had suggestions for improvement. For example, one user thought that a radial marking menu would be another good choice for menu configuration. Another mentioned that their answers might be different if there were a different number of buttons. We could therefore examine how the Shadow Button technique scales with the number of buttons.

We have presented Shadow Buttons, an approach to providing users with WIMP-based interface elements in pen-based applications. With Shadow Buttons, screen real estate is conserved by utilizing the hover state to invoke interface widgets, letting users enter ink anywhere on the screen, reclaiming space normally occupied by menus and toolbars. We explored the Shadow Button design space in the context of handwritten mathematical expressions and presented a informal usability study exploring users' perceptions of the interface technique. Our pilot study revealed that users, in

general, found the distance between the shadow region and the menu to be the most important factor in the usability of Shadow Buttons. In addition, users found the hover state to be a valuable approach to invoking the menu, validating the Shadow Button approach. Although our study was informal and there is a significant amount of future work to be done, we believe this work provides a useful starting point in developing and understanding Shadow Buttons in pen-based interfaces.

Acknowledgements

This worked is supported in part by Microsoft, IARPA, SAIC, and NSF CAREER award IIS-0845921. Thanks to the anonymous reviewers for their valuable suggestions.

References

[BZW*09] BRAGDON A., ZELEZNIK R., WILLIAMSON B., MILLER T., LAVIOLA JR. J. J.: Gesturebar: improving the approachability of gesture-based interfaces. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems* (New York, NY, USA, 2009), ACM, pp. 2269–2278.

[CHWS88] CALLAHAN J., HOPKINS D., WEISER M., SHNEIDERMAN B.: An empirical comparison of pie vs. linear menus. In *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1988), ACM, pp. 95–100.

[FKP*03] FITZMAURICE G., KHAN A., PIEKÉ R., BUXTON B., KURTENBACH G.: Tracking menus. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2003), ACM, pp. 71–79.

[GBH09] GROSSMAN T., BAUDISCH P., HINCKLEY K.: Handle flags: Efficient and flexible selections for inking applications. In *To appear: GI 2009 Conference Proceedings: the Graphics Interface Conference* (2009).

[GHB*06] GROSSMAN T., HINCKLEY K., BAUDISCH P., AGRAWALA M., BALAKRISHNAN R.: Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), ACM, pp. 861–870.

[GW00] GUIMBRETIÉRE F., WINOGRAD T.: Flowmenu: combining command, text, and data entry. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2000), ACM, pp. 213–216.

[KB94] KURTENBACH G., BUXTON W.: User learning and performance with marking menus. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1994), ACM, pp. 258–264.

[LZ04] LAVIOLA J., ZELEZNIK R.: Mathpad^2: A system for the creation and exploration of mathematical sketches. *ACM Transactions on Graphics 23*, 3 (Aug. 2004), 432–440. (Proceedings of SIGGRAPH 2004).

[ZMLL08] ZELEZNIK R., MILLER T., LI C., LAVIOLA JR. J. J.: Mathpaper: Mathematical sketching with fluid support for interactive computation. In *SG '08: Proceedings of the 9th international symposium on Smart Graphics* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 20–32.