HandyWidgets: Local Widgets Pulled-out from Hands

Takuto Yoshikawa, Buntarou Shizuki, and Jiro Tanaka
University of Tsukuba, Japan
{yoshikawa,shizuki,jiro}@iplab.cs.tsukuba.ac.jp

ABSTRACT

Large multi-touch tabletops are useful for collocated collaborative work involving multiple users. However, applying traditional WIMP interfaces to tabletops causes problems where users cannot reach GUI elements, such as icons or buttons, on the opposite side with their hands, and they sometimes have difficulty in reading the content of GUI elements because their view does not match the orientation of the content. To solve these problems, we present HandyWidgets that are widgets localized around users' hands. The widgets are quickly invoked by a bimanual multi-touch gesture which we call "pull-out". This gesture also allows users to adjust the position, orientation, and size of the widgets, in a continuous manner after invocation.

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces. - Interaction styles, Graphical user interfaces.

Author Keywords

multi-touch; tabletops; gestures; bimanual interaction; crossing.

INTRODUCTION

Large multi-touch tabletops are useful for collocated collaborative work involving multiple users. Users surround tabletops extend their hands and touch the screen at their location. However, applying traditional WIMP interfaces to tabletops causes problems where users cannot reach GUI elements, such as icons or buttons, on the opposite side with their hands, and they sometimes have difficulty in reading the content of GUI elements because their view does not match the orientation of the content. These problems impose extra actions on users, such as having to extend their hands to reach the desired elements, walking toward them, or tilting their heads to read the content. Furthermore, the first two actions might obstruct other users' line of sight or their territory.

Our approach to these problems is to localize the GUI elements around each user's hand as illustrated in Figure 1. This paper presents HandyWidgets, which are widgets (combinations of GUI elements) that are localized around users' hands. The widget are quickly invoked by a bimanual multi-touch gesture which we call *pull-out*. This gesture also allows users to adjust the position, orientation, and size of the widgets, in a continuous manner after invocation.

ITS'12, November 11–14, 2012, Cambridge, Massachusetts, USA.

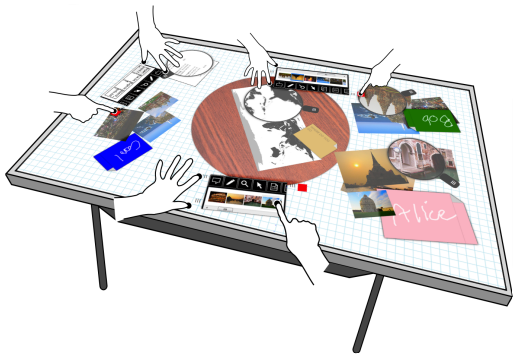

Figure 1. Overview of HandyWidgets. Each user can simultaneously display HandyWidgets.

RELATED WORK

Localization of widgets has previously been studied. In such research, where and how widgets appear and how users invoke widgets are important design factors and they have been intensively investigated.

Popup menus and localized widgets [1, 4, 5, 7–10, 12, 16] can be displayed at any position, but their orientation is fixed. This is because their designs assume users interact with a constant view, and they have not been designed for tabletops where multiple-users touch the screen at each location. Research such as [2, 3, 6, 13, 15, 17–20] allows users to not only determine the position of widgets, but also their orientation. Using orientation enables each user to match his/her view and the orientation of widgets.

In contrast, HandyWidgets' users can not only adjust the position and orientation of widgets, but also additional parameters, such as their size and type, through invocation gestures. Further, they can also adjust their parameters in a continuous manner by using the gestures, after they have invoked them.

On-screen buttons or physical buttons [7, 16] are wide-spread techniques of invocation. Stroke gestures have also been used as invocation techniques [1,12,19] . Further, multi-stroke gestures have been used. Benko et al. utilized a secondary finger to invoke a widget [4]. The number of touches has also been used for invocation [15, 20]. In addition, the rotation of pinch gestures has invoked widgets [9]. Additional states, which are provided by special devices, are also used for invocation. In [5, 8, 10] the hover states of styluses or hands have been used. Further, the contact shape of hands have invoked widgets [3, 6, 13, 18]. Guimbretière and Nguyen used the gesture of a hovering hand for invocation [11].

Figure 2. Basics of HandyWidgets. User a) touches tabletop with two fingers of non-dominant hand (base-fingers), b) places one finger of dominant hand (pull-finger) inside non-dominant hand, and drags pull-finger outside non-dominant hand to cross segment between base-fingers (base-segment), c) moves base-fingers and pull-finger to obtain desired position, orientation, and size of shown widget and d) releases pull-finger and executes command with widget using pull-finger.

HandyWidgets are invoked by a bimanual multi-stroke gesture combined with other gestures to avoid conflict with traditional gestures. HandyWidgets are robust to casual touching because users do not generally unintentionally perform this gesture. Moreover, this design requires no special devices and can be applied to conventional multi-touch screens that detect three or more coordinates of touches.

HANDYWIDGETS

HandyWidgets are invoked by a bimanual multi-touch gesture that we call pull-out. Widgets are displayed by using the following procedure (Figure 2), where the behavior of fingers in steps a) and b) is pull-out:

a) Touch the tabletop with two fingers of the non-dominant hand (base-fingers).

b) Place one finger of the dominant hand (pull-finger) inside the non-dominant hand, and drag the pull-finger outside the non-dominant hand to cross the segment between the base-fingers (base-segment).

c) Move the base-fingers and the pull-finger to obtain the desired position, orientation, and size of the shown widget.

d) Release the pull-finger, and execute a command with the widget using the pull-finger.

Once the pull-finger is lifted from the tabletop, the position, orientation, and size of the widget are fixed. Users can then interact with the widget. The widget is removed by releasing the base-fingers from the tabletop. This design provides ad-hoc use of HandyWidgets. However, if users need to keep the widget visible even after their fingers have been released, a pin button on the widget allows it to be pinned to the tabletop. This design allows users to interact with the widget with both hands in a relaxed way. In addition, it is possible to simultaneously display two or more widgets.

Features

Fluidity. Pull-out does not conflict with traditional multi-touch gestures, such as pinch or swipe, and can coexist with them. This means that pull-out represents a new interaction vocabulary. Thus, as seen in Figure 3, users can invoke a widget and fluidly execute commands with their dominant hand while using pinch gestures with their non-dominant hand to adjust the scale, rotation, or translation of objects.

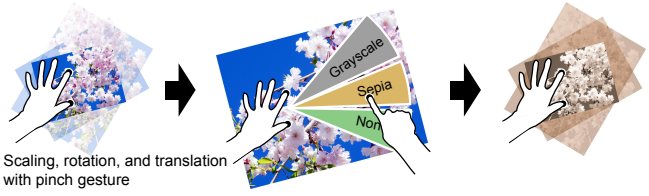

Figure 3. Fluid manipulation of objects. Note that non-dominant hand keeps pinching.

Robustness against casual touching. Pull-out is distinctive, that is, users do not generally perform this gesture unintentionally. Therefore, HandyWidgets are hard to be accidentally invoked by casual touching.

Occlusion aware design. The design of HandyWidgets and pull-out gestures avoids occlusion in a similar manner to those found in previous works [6, 13, 18, 19], where the interaction technique itself has been designed to avoid the widget from being occluded. The hands that invoke widgets in HandyWidgets do not occlude the widgets because they appear between the hands. Because dragging the base-fingers can also adjust the position, orientation, and size of the widgets, it helps users to invoke widgets even if they are near the edge of the screen (Figure 4).

Figure 4. Showing widget near the edge of the screen.

ADVANCED INTERACTION DESIGN

This section presents advanced interaction design that enhances the basics of HandyWidgets we previously described.

Pull-in. Dragging an object with the pull-finger to cross base-segments can be used as a gesture (Figure 5). We called this gesture *pull-in*. Users can copy objects to their clipboards with pull-in, and they can then choose a copied object from the widget invoked by pull-out and paste it.

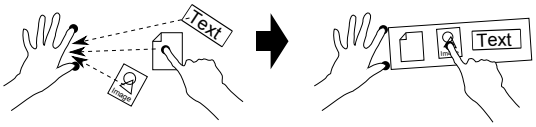

Figure 5. Pull-in.

Object specific widget. While pull-out on the background can be assigned to invoke a general widget, pull-out on a foreground object can be assigned to invoke its specific widget. It is possible in this way to display an effect menu on an image while a tool set is being displayed on the background, for example.

Various pull-outs. Various pull-outs are available. This enables users to select types of widgets corresponding to each pull-out. For example, the additional use of single crossing with two pull-fingers (Figure 6) or double crossing with one pull-finger (Figure 7) enables users to display a system menu with the former and a keyboard with the latter, while basic pull-out invokes a tool set. Thus, users can invoke various widgets using various pull-outs depending on their purposes.

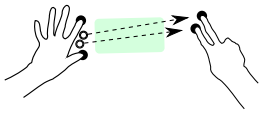

Figure 6. Single crossing with two pull-fingers.

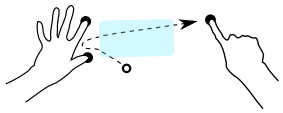

Figure 7. Double crossing with one pull-finger.

Figure 8. Widget that transforms from Numeric keypad into QWERTY keyboard.

Dynamic widgets. Users can dynamically transform the structure of a widget according to the positional relation of a base-segment and a pull-finger, which determines the position, orientation, and size of the widget. This enables adaptive use of the widget depending on the required functions. Figure 8 shows such usage, where the widget transforms from a Numeric keypad into a QWERTY keyboard. While users can input numbers quickly with the former, the latter is available for long texts.

EXPLORING DESIGN SPACE

Pull-out gestures not only allow users to determine the position, orientation, and size of widgets, but they also add a degree of freedom to the design space of widgets. We implemented four *widget models* to explore the design space of HandyWidgets. Each model is outlined in Figure 9.

The Drawer Model (Figure 9a) is the model in which items of the widget are displayed from one side according to the length between the base-segment's center and the pull-finger, which we called the distance-pulled. In this model, placing frequently used items on one side of the widget allows fast access to the items, while providing on-demand access to the items rarely used by opening it further. This model is suitable for a tool palette in which frequency of use differs greatly between items, such as those in painting or CAD applications.

The Popup Model (Figure 9b) is the model in which the widget is immediately displayed when pull-out is performed, regardless of the distance-pulled. Although this model causes occlusion by the user's hand, it minimizes the time to access the widget. The model is thus suitable for widgets that only include some frequently-used items. We used the model for pie-menus, which can be used quickly without depending on visual cues once users have become familiar with them.

The Scale Model (Figure 9c) is the model in which the widget is scaled according to the distance-pulled. In this model, users can adjust the size of widgets, which is needed to determine their area on the screen, by using pull-out. The model is suitable for widgets that require definitions of areas, such as zoom lenses or shape tools.

The Transform Model (Figure 9d) is the model in which the structure of the widget dynamically transforms according to the distance-pulled. This model is suitable for widgets in which users want to modify usage depending on their purpose such as that illustrated in Figure 8.

While these four models only utilize the distance-pulled, more parameters are available, e.g., the rotation and length

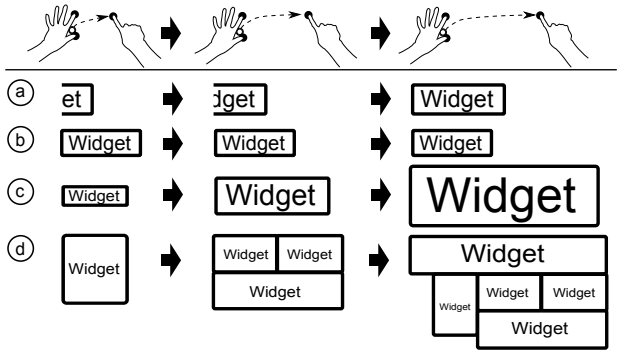

Figure 9. Four models of widgets we implemented.

of a base-segment. This indicates that HandyWidgets have a broad design space.

IMPLEMENTATION

Invoking HandyWidgets needs to recognize the crossings of pull-fingers through base-segments. Base-segments are defined to fit the users' hand shapes shown in Figure 10. This technique can be applied to conventional multi-touch screens that detect three or more coordinates of touches. We describe the recognition technique below.

First, the system tries to find the nearest touch point within a threshold distance (15 cm in our current implementation) for each touch point. If such points are found, then the system defines the segment between them as a base-segment. These segments are updated dynamically. Second, the crossing of pull-finger is detected. The system regards touch points that are within a half length of the base-segment from its center as pull-fingers. Touches farther than the threshold from the other touches are also regarded as pull-fingers. Next, crossing through the base-segments of pull-fingers is detected. After that, the system estimates users' locations. Lines through base-segments roughly point to users when they place their index finger and thumb on the tabletop as base-fingers. Based on this observation, our system defines the nearest intersections of the lines with four sides of the screen as users' locations. Finally, the system determines which hand has pulled out the widgets by using the users' locations. If a pull-finger has crossed a base-segment from left to right from the viewpoint of each user's location, then our system determines the right hand has pulled out the widget, and vice versa. This allows users to pull-out widgets from both right and left hands.

HandyWidgets are rendered within the rectangular area between the base-segment and the pull-finger. Our system defines the side of the area nearest to the users' location to be the widgets' floor to display the widgets. This makes their orientation suit the users' view.

There have been some multi-touch systems that have used the segment defined by users (e.g., [20]), but we utilized the segments as targets of crossing.

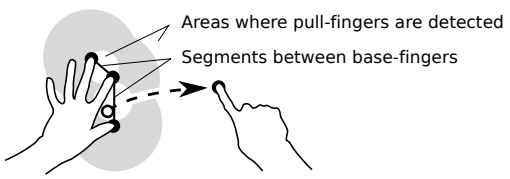

Figure 10. Definitions of base-segments, and detection of pull-finger.

APPLICATION

We developed an application (Figure 1), similar to [14], that supports collocated collaborative work such as discussions or meetings. In this application, each user can annotate images or text documents. Users can invoke HandyWidgets at any position and orientation, and they display images or text documents in a shared folder with file browsers embedded in the widgets. Annotations by painting or texts, and zoom lenses are also available with the widgets. Users can scale, rotate, or translate objects. Objects including images, texts, and paintings can be stored in a shared clipboard by using pull-in.

Object specific widgets are available. Pull-out with one finger single crossing on images or annotations invokes a pie-menu of the Popup Model, which determines image effects, or background colors. Moreover, pull-out with one finger single crossing on the background invokes a tool set of the Drawer Model; pull-out with two fingers single crossing on the background invokes a system menu of the Drawer Model; pull-out with one finger double crossing on the background invokes a keyboard (Figure 8) of the Scale and Transform Model.

EXHIBITION

We demonstrated a prototype system at an open house at our university. We used a 60-inch tabletop, and prepared a simple diagram-drawing application with HandyWidgets. Twenty three participants used the system and their behaviors were recorded during the exhibition. Participants used the system simultaneously with one of the authors. Many participants failed to invoke the widget at first because they did not know how far they should place the pull-finger between base-fingers. However, they immediately learned how to do it after being instructed to place their finger further. This observation may indicate the ease of learning HandyWidgets. Still, accidental invocations were observed when users quickly swiped more than three fingers, and invocation failures were also observed. However, these problems seemed to be due to failures in finger detection by the optical multi-touch display we used, because when we carried out tests with a smaller, more accurate multi-touch display, no problems were observed.

DISCUSSION

Fatigue is an issue that needs to be further studied because pull-out gestures involve three fingers, although we did not observe any fatigue in users at the exhibition. Our system allows users to touch the tabletop with more than two fingers of their non-dominant hand, thus they can stably pull-out and use widgets with less fatigue by allowing their non-dominant hand touch with all fingers for invocation. Further, allowing users to place their palms in contact with the tabletop will also reduce fatigue.

CONCLUSIONS AND FUTURE WORK

We presented HandyWidgets, which represented widgets localized around hands. The widgets are invoked by a distinctive bimanual multi-touch gesture, called pull-out, which can coexist with traditional gestures and is hard to be accidentally invoked by casual touches. Moreover, various pull-outs can be defined, each of which can be used to invoke their corresponding widgets.

One direction for future work is to explore the properties of pull-out. We will use more parameters, such as the rotation and length of base-segments, to add more degrees of freedom to widgets. Another direction we plan is to utilize the geometry of hands. This will realize four widgets corresponding to four base-segments defined by five fingers. It can thus allow users to relate spatial memory to widgets. Moreover, arranging the elements of a widget along the strokes, similiar to [17], of pull-out achieves ergonomical design, because the strokes follow the range of motion of one's arm and hand. We could, for example, develop a curved tool set or keyboard.

REFERENCES

1. G. Apitz and F. Guimbretière. CrossY: A crossing-based drawing application. In *UIST '04*, pp. 3–12.
2. N. Banovic, F. C. Y. Li, D. Dearman, K. Yatani, and K. N. Truong. Design of unimanual multi-finger pie menu interaction. In *ITS '11*, pp. 120–129.
3. T. Bartindale, C. Harrison, P. Olivier, and S. E. Hudson. SurfaceMouse: supplementing multi-touch interaction with a virtual mouse. In *TEI '11*, pp. 293–296.
4. H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06*, pp. 1263–1272.
5. A. Bragdon, R. DeLine, K. Hinckley, and M. R. Morris. Code Space: touch + air gesture hybrid interactions for supporting developer meetings. In *ITS '11*, pp. 212–221.
6. P. Brandl, J. Leitner, T. Seifried, M. Haller, B. Doray, and P. To. Occlusion-aware menu design for digital tabletops. In *CHI EA '09*, pp. 3223–3228.
7. J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. An empirical comparison of pie vs. linear menus. In *CHI '88*, pp. 95–100.
8. G. Fitzmaurice, A. Khan, R. Pieké, B. Buxton, and G. Kurtenbach. Tracking Menus. In *UIST '03*, pp. 71–79.
9. D. Freeman and R. Balakrishnan. Tangible Actions. In *ITS '11*, pp. 87–96.
10. T. Grossman, K. Hinckley, P. Baudisch, M. Agrawala, and R. Balakrishnan. Hover Widgets: using the tracking state to extend the capabilities of pen-operated devices. In *CHI '06*, pp. 861–870.
11. F. Guimbretière and C. Nguyen. Bimanual marking menu for near surface interactions. In *CHI '12*, pp. 825–828.
12. K. Hinckley, P. Baudisch, G. Ramos, and F. Guimbretiere. Design and analysis of delimiters for selection-action pen gesture phrases in Scriboli. In *CHI '05*, pp. 451–460.
13. K. Hinckley, K. Yatani, M. Pahud, N. Coddington, J. Rodenhouse, A. Wilson, H. Benko, and B. Buxton. Pen + Touch = New Tools. In *UIST '10*, pp. 27–36.
14. U. Hinrichs, S. Carpendale, and S. D. Scott. Evaluating the effects of fluid interface components on tabletop collaboration. In *AVI '06*, pp. 27–34.
15. K. Kurihara, N. Nagano, Y. Watanabe, Y. Fujimura, A. Minaduki, H. Hayashi, and Y. Tutiya. Toward localizing audiences' gaze using a multi-touch electronic whiteboard with spiemenu. In *IUI '11*, pp. 379–382.
16. G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *CHI '94*, pp. 258–264.
17. D. Leithinger and M. Haller. Improving menu interaction for cluttered tabletop setups with user-drawn path menus. In *TABLETOP '07*, 121–128.
18. J. Rekimoto. SmartSkin: An infrastructure for freehand manipulation on interactive surfaces. In *CHI '02*, pp. 113–120.
19. V. Roth and T. Turner. Bezel Swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *CHI '09*, pp. 1523–1526.
20. S. Strothoff, D. Valkov, and K. Hinrichs. Triangle Cursor: interactions with objects above the tabletop. In *ITS '11*, pp. 111–119.