

# Spatial Arrangement of Data and Commands at Bezels of Mobile Touchscreen Devices

Toshifumi Kurosawa, Buntarou Shizuki, and Jiro Tanaka

University of Tsukuba, Japan.  
{kurosawa, shizuki, jiro}@iplab.cs.tsukuba.ac.jp

**Abstract.** We show a data and commands arrangement design on mobile touchscreen devices. In this design, a user can arrange any data, such as text and Web pages, *at the bezel* of the touchscreen by using a simple crossing gesture across the bezel. Our design has three main merits: data can be arranged while the small display area on mobile environment is kept open; the user can continuously execute multiple commands with the user’s minimal visual attention; and memorizing the locations of the data is made easier by utilizing the user’s spatial memory.

**Keywords:** Data placing, data management, touch gestures, bezel gestures, shortcuts, menu, crossing, spatial memory.



Fig. 1: In the case of the proposed design, the user uses a double-crossing gesture, named a “bezel check,” which is performed across the bezel of a touchscreen device, just like drawing a check mark. The yellow arrow indicates a swipe movement on the touchscreen and its bezels. (Left) When the user copies the selected text by a bezel check, (Center) the copied datum is stored in a virtual clipboard placed at the bezel where the gesture was given. The virtual clipboard then appears as a semi-translucent green rectangle. (Right) The user can use (e.g., paste or Web search) a copied datum by using a marking menu displayed by swiping from the bezel to the corresponding virtual clipboard.

## 1 Introduction

Many designs of user interfaces enabling a user to arrange thumbnails of documents spatially in a manner that allows many documents held on computers to be managed, called *data arrangement designs*, have been explored. Prominent examples of such designs are DataMountain [1] and BumpTop [2]. A great advantage of such data-arrangement

designs is that they allow the user to organize the documents by utilizing his or her spatial memory, thereby enabling the user to group related documents. While the efficacy of such designs in environments in which a large amount of screen real estate is available, such as desktop environments [1–3] and multiple displays environment [4, 5], was demonstrated, data-arrangement designs in mobile environments in which the amount of screen real estate is limited are still open.

In this paper, we show a data arrangement design on mobile touchscreen devices. In the case of this design, a user can arrange any data, such as texts and Web pages, *at the bezels* of a touchscreen by using a simple double-crossing gesture across the bezels, named *bezel check*. After the user performs the bezel check, the system generates a virtual clipboard to store the datum, which is determined by the context. If the user wants to use a datum in a virtual clipboard, he or she selects an intended command from a marking menu, whose items depend on the context, displayed by swiping from the bezel to the corresponding virtual clipboard.

Moreover, the user can also arrange commands such as text search at a bezel in the same way as the above-described data arrangement. To do this arrangement, a user firstly selects a command by using an ordinary method (e.g., select a “search” command from a menu bar of a text editor) and then performs a bezel check. After that, a virtual clipboard is generated, which serves as a *shortcut* to the command. Thereafter, the user can execute the command arranged at the bezel by swiping from the bezel. If the command needs an argument such as a text, the user firstly selects the text to be an argument of an intended command. The user then drags it to the bezel where the command was arranged and performs a double-crossing across the bezel.

In the proposed design, data and commands are arranged in physical space on a touchscreen; the design is thus helpful for memorizing the places storing the data and commands because it utilizes a user’s spatial memory while keeping the small display area of a mobile touchscreen open. Moreover, the user can execute command(s) repeatedly and continuously with the user’s minimal visual attention because a command arranged at the bezel is executed by a simple crossing gesture, which promotes the fluid composition of commands [6], and is easier than a pointing to select a target [7].

## 2 Related Work

Many data-arrangement designs have been proposed [1–3, 5, 8, 9]. The “pile metaphor,” a pioneering one of these designs, was proposed more than a few decades ago [3], and since then it has been applied to various uses (e.g., organizing files on a 3D desktop [2], organizing physical and digital documents on tabletops [9], and even organizing small displays showing digital documents [5]). Data Mountain [1], which allows a user to arrange digital documents at arbitrary positions on an inclined plane in a 3D desktop environment by using a simple 2D interaction technique. They showed that the spatial layout created by Data Mountain can utilize users’ spatial memory. While the proposed design also adopts a spatial layout, it is different from the above-mentioned designs; namely, data and even commands are arranged only at bezels on a touchscreen. Thus, placing data and commands does not consume real estate of touchscreens, and it keeps the small display area of mobile touchscreen devices open.

Many bezel gestures have been proposed [10–13] and used [14–16]. Bragdon et al. found that bezel-initiated gestures were the fastest and the most preferred gestures in a mobile environment [17]. Serrano et al. proposed Bezel-Tap Gestures [12], which allows a user to immediately execute a command on a handheld tablet device regardless of whether the device is alive or in sleep mode. Roth et al. showed that a bezel-initiated swipe supports multiple selections, cut, copy, paste, and other operations without conflicting with pre-defined multi-touch gestures such as zooming, panning, and tapping [13]. Wagner et al. proposed a design space called BiTouch [18], which introduces support commands in a kinematic chain model for interacting with handheld tablet devices (including their bezels). In contrast, the design proposed in the current work uses bezel gestures to arrange data and apply commands to data. This design makes it possible to design bezel gestures that allow a user to arrange data spatially and to execute commands with the user’s minimal visual attention.

Many researchers have explored crossing gestures on the touchscreen environment and revealed its efficacy [7, 13, 19–21]. Moreover, many crossing-based interaction techniques, such as Control Menu [22], FlowMenu [23], and CrossY [24], and Bezel Swipe [13], have been proposed. These crossing-based gestures and techniques allow the user to execute multiple commands continuously. In the case of the proposed design, the user also uses crossing gestures to execute multiple commands continuously.

### 3 A Design for Arranging Data and Commands

#### 3.1 Arranging Data and Commands

In the proposed design, a user performs a double crossing gesture [25] across a bezel of a touchscreen, which is our own bezel-initiated gesture we named a *bezel check*. This gesture is designed to allow the user to arrange data and commands spatially with the destination being indicated by the simple gesture. First, the user selects a datum or a command by using an ordinary method (e.g., a long tap to select a text and select a “search” command in a menu bar). Next, the user performs a bezel check: swiping from the bezel across part of the display edge into the interior of the screen and returning the finger to the outside of the screen, just like drawing a check mark (Fig. 1 left). Then, the system generates a virtual clipboard (a semi-translucent green rectangle in the center of Fig. 1) at the bezel where the bezel check is completed, and it stores the selected datum or command in the virtual clipboard.

#### 3.2 Using the Arranged Data and Commands

To use the datum in the virtual clipboard, the user begins with activating an application that uses the datum. The user then selects a command in the marking menu, whose items depend on the activated application, displayed by swiping from the bezel to the corresponding virtual clipboard (Fig. 1 right).

A command in the virtual clipboard can be executed in two ways. The first way is for the user to swipe from the bezel to the corresponding virtual clipboard to execute a command with no argument. In this case, the user selects the command in the marking

menu displayed near the virtual clipboard. The second way is used when the user wants to give an argument to a command. For example, if the user wants to search a text in an Web page, the user firstly selects the text in the page (Fig. 2 left) or in another virtual clipboard by a gesture called “holding,” which will be described later (Fig. 3). Next, the user drags the selected text to the virtual clipboard storing the searching command and performs a double crossing across the clipboard as shown in the center of Fig. 2. The search command is then executed with the selected text. The user also executes the command repeatedly by repeating bezel crossings in a manner just like drawing circles (Fig. 2 right).

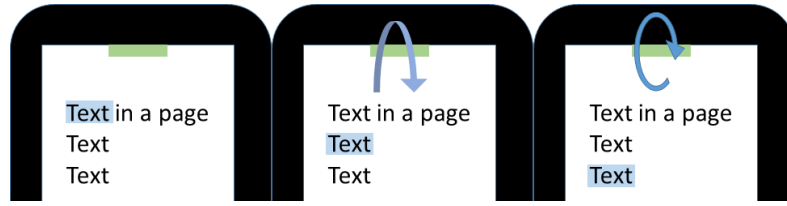


Fig. 2: Executing a search command with an argument. (Left) Selecting a text. (Center) Dragging the selected text to the virtual clipboard that stores a searching command and performing a double crossing across the clipboard from the interior of the screen. (Right) Executing the command repeatedly by repeating bezel crossings in a manner just like drawing circles.

### 3.3 Organizing Virtual Clipboards

Users can organize virtual clipboards by moving, integrating, and deleting them.

*Moving.* A user can move a virtual clipboard to any part of a bezel of the screen. To do this, the user first “holds” the clipboard by swiping from the bezel to the clipboard and then tapping the clipboard with another finger (Fig. 3). Next, the user moves the clipboard by dragging it to the free parts of the bezels (i.e., parts of the bezels where no virtual clipboard is placed) using a finger. This procedure allows the user to easily edit positional relations among virtual clipboards. For example, the user can arrange the data in chronological order.

*Integrating.* The user can integrate a virtual clipboard into another one. When the user moves the clipboard to another one, the stored data in the two clipboards are stored in the destination clipboard of dragging (Fig. 5). In addition, if the user touches a clipboard that holds multiple data, the hierarchy of the data is expanded (left of Fig. 6). To use an object in the hierarchy, the user selects the object by swiping from the bezel to the object. This procedure allows the user to easily group many data, such as login data and mail addresses, into categories.

*Deleting.* The user can delete a virtual clipboard by moving it to a trash can displayed at the center of the screen. The trash can appears there when the user holds a virtual clipboard.

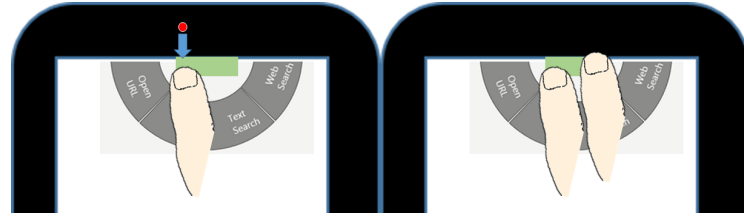


Fig. 3: Holding a virtual clipboard by swiping from the bezel to the clipboard and then tapping the clipboard with another finger.

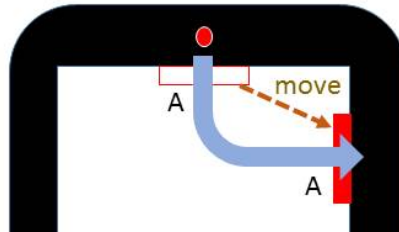


Fig. 4: If the user drags a virtual clipboard to a free space of a bezel, the clipboard moves there.

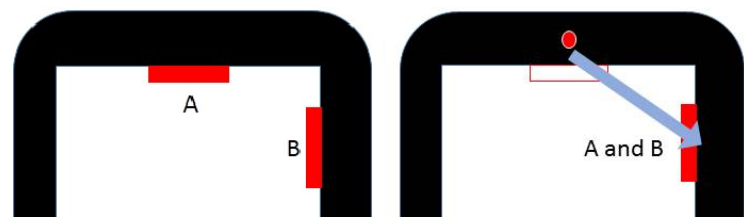


Fig. 5: If the user drags the clipboard containing *A* to the clipboard containing *B*, the two clipboards are integrated to have two contents, *A* and *B*.

### 3.4 Merits

The proposed design allows the user to use the bezel of a touchscreen to arrange data and commands. It thereby leads to the following three merits.

*Keeping the display area open.* The user can store data and commands while keeping the (small) display area of a mobile touchscreen device open, because the proposed design only uses the bezel of a touchscreen. In addition, the marking menus displayed by a single swipe from the bezel are a kind of bezel menus, which enable interaction with a mobile touchscreen with the user's minimal visual attention and thus solve the occlusion problem [26].

*Executing multiple commands continuously.* The user can continuously execute multiple commands because the execution requires only simple crossing gestures, which promote fluid composition of commands [6] in the same manner as reported in [23, 24, 27]. Moreover, the user can execute a command repeatedly by performing a bezel check continuously across the virtual clipboard that stores the command, just like drawing circles on the bezel (Fig. 2 right).

*Mapping data to place.* The user can arrange data and commands spatially. For example, the user can arrange placed data in the order of priority or chronologically. This spatial layout of data might be helpful in memorizing the data's places by utilizing the user's spatial memory [1, 3, 4].

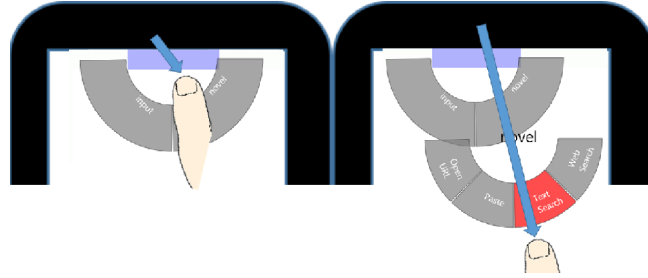


Fig. 6: (Left) Expanding the multi-objects-clipboard by swiping. (Right) Selecting the content.

## 4 Prototype Applications and their Use Cases

A prototype of the proposed design was developed as a system daemon running on a Dell 10.1-inch Windows 8 mobile touchscreen device. The daemon continuously monitors the active application, since the marking menu (displayed by swiping from the bezel) and the command executed when a user performs a bezel check depend on the active application, as shown in Table 1. These applications and their use cases are described in the following sections.

#### 4.1 Data Arrangement

As for the prototype design, texts and Web pages can be the data to be stored at bezels. As a result, the user can use the bezel as a multiple clipboard or bookmark bar. A use case of such a data arrangement is described as follows. If the user wants to place a text at a bezel, he or she firstly selects the text and performs a bezel check. Then, the daemon generates a virtual clipboard where the bezel check is performed and copies the text to it. After that, if the user performs a swipe from the bezel across the virtual clipboard when the text editor is active, the daemon displays the “paste” and “text search” command in the marking menu, as shown in Fig. 7. Another use case is that if the user performs a bezel check when an Web browser is active, the opened URL is bookmarked at the virtual clipboard.

Table 1: Commands displayed in the marking menu at the time each application is active.

Application	On a bezel check	Commands
Text editor	Copy	Paste / Text Search
Web browser	Copy / Bookmark	Paste / URL open Web search / Text search
PDF viewer	Copy	Text search

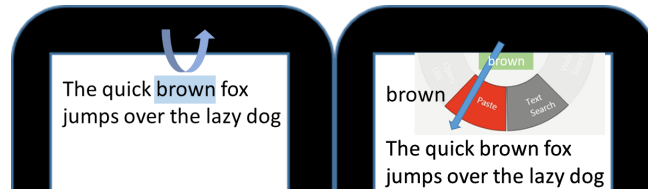


Fig. 7: Using a virtual clipboard with a text editor. (Left) Copying a text. (Right) Display a marking menu at the virtual clipboard.

#### 4.2 Commands Arrangement

We implemented some arrangeable commands, namely, searching, changing volume, and page transition. Use cases of arranging commands by using a search command are described as follows. When an Web browser or a text editor is activated and the user executes a bezel check, the virtual clipboard that contains the search command is generated at the bezel where the bezel check is completed. If the user performs a swipe from the bezel and a crossing across the virtual clipboard when an application such as a

text editor or an Web browser is activated, the daemon displays the “search” command in the marking menu. After the user selects the command, the daemon activates the search command, just like executing a keyboard shortcut *Ctrl + F*. Then, the user can execute the search by typing a search word or pasting a copied word.

The user can also execute such commands with an argument. For example, if a user selects a text in a text editor and drags it to a virtual clipboard storing a search command, and then performs a double-crossing across the clipboard, the user can execute the search command with the selected text as the argument, as shown in the center of Fig. 2. Moreover, in this case, the user can find occurrences of the text one by one (i.e., incrementally search the text) by executing the search command repeatedly by double crossing the bezel repeatedly, as shown in Fig. 2 right.

In addition, we have also developed a drawing tool similar to CrossY [24] which provides the user with our design. This tool revealed that the continuous execution of commands is especially useful. For example, after selecting a drawing object, the user can continuously execute the commands “changing line width,” “changing line color,” and “swap to the top layer” with only one stroke gesture. Moreover, the commands can be combined into one virtual clipboard by integrating multiple clipboards, each of which stores a command as described in the section “Organizing the Virtual Clipboards.” The user can thus define a complicated command with two or more processes such as “Increase the line width while changing the line color of an argument object to red.”

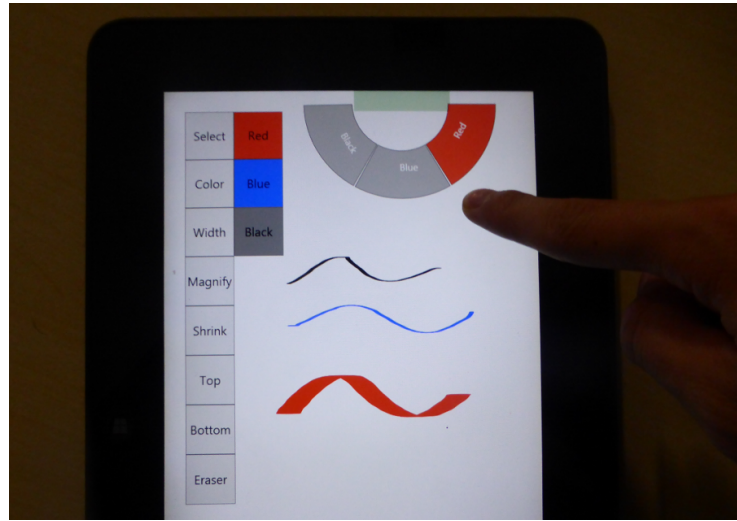


Fig. 8: A drawing tool which provides the user with our design.



## 5 Implementation

The prototype system daemon was implemented as a separate program from the applications for providing application-independent services.

To detect double-crossing gestures across the bezel and swipes from the bezel, touch-sensitive transparent windows were placed near the four edges of the bezels, as shown in Fig. 9 left.

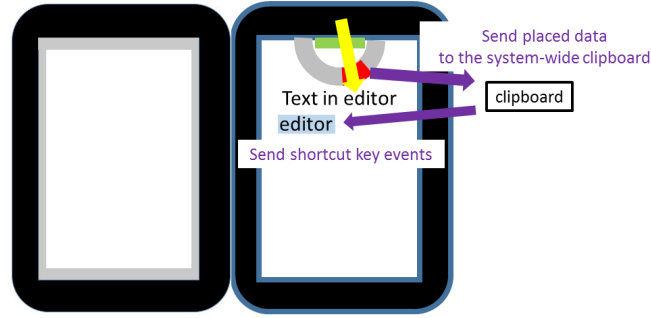


Fig. 9: (Left) Four touch-sensitive transparent windows (gray). (Right) Send data to an active application.

The daemon monitors an active application to transmit the data from it. The transmission process is illustrated on the right of Fig. 9. To transmit the data between an active application and the virtual clipboard, the daemon sends key events and uses the system-wide clipboard provided by the operating system as a relay point. For example, when the user activates a text editor and performs a bezel check, the daemon sends a *Ctrl + C* key event in order to copy the selected text to the system-wide clipboard. The daemon then retrieves the copied data from the clipboard. When the user selects a command in the marking menu displayed by swiping from the bezel, the daemon also sends the data contained in the virtual clipboard to the system-wide clipboard, and then it sends appropriate shortcut-key events on the active application (e.g., send *Ctrl + V* to paste).

## 6 Conclusion and Future Work

We show a data and commands arrangement design of mobile touchscreen devices. The design uses virtual clipboards, which are laid out along the bezel of the touchscreen (so they help in memorizing the contents by utilizing the user's spatial memory) and are generated by a simple crossing gesture across the bezels, bezel check. The data and commands are arranged at the bezel, so the display area of a mobile touchscreen is kept open. Moreover, users can execute command(s) repeatedly and continuously because a command displayed at the bezel is executed by a crossing gesture.

For future work, we will evaluate our design in terms of the effects of the spatial layout of the proposed design on the user's cognitive processing [1, 3, 4].

## References

1. G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich. Data Mountain: using spatial memory for document management. In *Proc. UIST '98*, pp. 153–162.
2. A. Agarawala and R. Balakrishnan. Keepin' it real: Pushing the desktop metaphor with physics, piles and the pen. In *Proc. CHI '06*, pp. 1283–1292.
3. R. Mander, G. Salomon, and Y. Y. Wong. A pile metaphor for supporting casual organization of information. In *Proc. CHI '92*, pp. 627–634.
4. E. D. Ragan, A. Endert, D. A. Bowman, and F. Quek. The effects of spatial layout and view control on cognitive processing. In *CHI EA '11*, pp. 2005–2010.
5. A. Girouard, A. Tarun, and R. Vertegaal. DisplayStacks: interaction techniques for stacks of flexible thin-film displays. In *Proc. CHI '12*, pp. 2431–2440.
6. P. Baudisch. Don't click, paint! using toggle maps to manipulate sets of toggle switches. In *Proc. UIST '98*, pp. 65–66.
7. J. Accot and S. Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proc. CHI '02*, pp. 73–80.
8. N. Watanabe, M. Washida, and T. Igarashi. Bubble Clusters: An interface for manipulating spatial aggregation of graphical objects. In *Proc. UIST '07*, pp. 173–182, 2007.
9. J. Steimle, M. Khalilbeigi, and M. Mühlhäuser. Hybrid groups of printed and digital documents on tabletops: a study. In *CHI EA '10*, pp. 3271–3276.
10. K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz. Sensing techniques for mobile interaction. In *Proc. UIST '00*, pp. 91–100.
11. S. Kim, J. Yu, and G. Lee. Interaction techniques for unreachable objects on the touchscreen. In *Proc. OzCHI '12*, pp. 295–298.
12. M. Serrano, E. Lecolinet, and Y. Guiard. Bezel-Tap gestures: quick activation of commands from sleep mode on tablets. In *Proc. CHI '13*, pp. 3027–3036.
13. V. Roth and T. Turner. Bezel Swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proc. CHI '09*, pp. 1523–1526.
14. R. Zeleznik, A. Bragdon, F. Adeptura, and H. Ko. Hands-On Math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proc. UIST '10*, pp. 17–26.
15. K. Hinckley, K. Yatani, M. Pahud, N. Coddington, J. Rodenhouse, A. Wilson, H. Benko, and B. Buxton. Pen + touch = new tools. In *Proc. UIST '10*, pp. 27–36.
16. N. Yu, D. Huang, J. Hsu, and Y. Hung. Rapid selection of hard-to-access targets by thumb on mobile touch-screens. In *Proc. MobileHCI '13*, pp. 400–403.
17. A. Bragdon, E. Nelson, Y. Li, and K. Hinckley. Experimental analysis of touch-screen gesture designs in mobile environments. In *Proc. CHI '11*, pp. 403–412.
18. J. Wagner, S. Huot, and W. Mackay. BiTouch and BiPad: designing bimanual interaction for hand-held tablets. In *Proc. CHI '12*, pp. 2317–2326.
19. P. Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. UIST '04*, pp. 193–196.
20. Y. Luo and D. Vogel. Crossing-based selection with direct touch input. In *Proc. CHI '14*, pp. 2627–2636.
21. C. Chen, S. T. Perrault, S. Zhao, and W. T. Ooi. BezelCopy: An efficient cross-application copy-paste technique for touchscreen smartphones. In *Proc. AVI '14*, pp. 185–192, 2014.

22. S. Pook, E. Lecolinet, G. Vaysseix, and E. Barillot. Control Menus: Execution and control in a single interactor. In *CHI EA '00*, pp. 263–264.
23. F. Guimbretière and T. Winograd. FlowMenu: Combining command, text, and data entry. In *Proc. UIST '00*, pp. 213–216.
24. G. Apitz and F. Guimbretière. CrossY: A crossing-based drawing application. In *Proc. SIGGRAPH '05*, pp. 930–930.
25. T. Nakamura, S. Takahashi, and J. Tanaka. An object selection technique using hand gesture in large display -proposing double-crossing and comparing with other techniques. *IEICE Trans.*, Vol. J96-D, No. 4, pp. 978–988. (In Japanese).
26. M. Jain and R. Balakrishnan. User learning and performance with bezel menus. In *Proc. CHI '12*, pp. 2221–2230.
27. B. Shizuki, T. Hisamatsu, S. Takahashi, and J. Tanaka. Laser pointer interaction techniques using peripheral areas of screens. In *Proc. AVI '06*, pp. 95–98.