Generalized Eyes-free Interaction for Use with Large Displays

Jens Bauer, Achim Ebert
Computer Graphics and HCI Lab, University of Kaiserslautern
Postbox 3049, 67653 Kaiserslautern, Germany
{j_bauer,ebert}@cs.uni-kl.de

Oliver Kreylos, Bernd Hamann
Institute for Data Analysis and Visualization, Department of Computer Science, University of California, Davis
One Shields Ave Davis, CA 95616, U.S.A.
{okreylos,hamann}@cs.ucdavis.edu

ABSTRACT

Smart phones and tablet computers are becoming increasingly ubiquitous and powerful. They can serve as replacements for currently used input devices, and provide novel functionality not achieved with traditional devices. Furthermore, their ubiquity ensures that they scale well to multi-user environments, where users can use their own devices.

Several attempts have already been made to use smart phones and tablets as input devices, but all of these have been one-shot and problem-specific. We present an application-independent way to integrate smart phones and tablets into existing systems, using a rapid development process. This approach is based on Marking Menus, but extends the basic idea by employing the special capabilities of current consumer-level smart phones and tablets.

Author Keywords

input devices; collaborative interaction; multimodal interaction; 3D Interaction

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces

General Terms

Human Factors

INTRODUCTION

Smart phones and tablet computers are becoming increasingly popular and powerful. Current consumer-level devices feature multi-core CPUs and dedicated GPUs. They contain multi-touch screens, GPS receivers, compasses, and accelerometers, and offer connectivity via WiFi, bluetooth, and 3G technologies. Given those capabilities, smart phones and tablets are tempting alternatives to "traditional" input devices, particularly for VR environments. Smart phones and tablets offer several benefits, due to their ubiquity and added functionality compared to typical input devices.

As these devices are multi-functional and wide-spread, most users will already own at least one, which precludes the need to buy often expensive single-purpose input devices. The same holds for multi-user environments, where each user can use her own device to control the system. Due to the devices' portability, they could potentially be used as "mobile memory" to transfer data between environments.

Currently, to control a remote system, such as a PC or a display wall, etc., with a mobile device the traditional approach is to transfer a part of the displayed screen to the screen of the mobile device and allow touch interaction with that part. Another often used technique is to present a number of standard UI controls to the user that will alter the state of the remote application. These methods incur some common problems. The biggest problem is that users will have to divert their attention from the main application onto the display of the mobile device in order to interact. For the screen-replication-approach there is the additional problem of the small screen real estate on mobile devices, leading to the situation that often only parts of an application can be shown at once. This makes it necessary to provide navigation controls to users. The so-called "Fat-Finger-Problem" can also cause problems in this scenario, where users might accidentally invoke a function they did not mean to use.

Using standard touch-UI elements, like the ones provided by current smartphone and tablet operating systems will most often lead to the point, where changes to the functionality of the host system will cause the layout of a mobile device UI to change in order to accommodate the new functionality. This might disorient users and also causes considerable work overhead to change the mobile application.

We propose to utilize *Marking Menus* [7], a radial menu structure, as a central element of a novel interaction method employing the multi-touch screen of mobile devices. The touch screen is used to show (hierarchical) radial menus as they pop up. This enables eyes-free interaction for experienced users who do not need the visual feedback from the mobile device, and leads to increased efficiency for those users, while at the same time keeping the menu structures visible should they be needed. This selection method is extended by the usage of tracking sensors, accelerometers, multi-touch

PID-MAD 2013, Aug 27 2013, Munich, Germany.
(In Conjunction with MobileHCI '13, Aug 27-30 2013, Munich, Germany.)

and/or in-menu slider controls to provide a larger variety of interaction possibilities, which are explained in detail later in this paper.

The advantages of this design over existing interaction methods are the following:

- Eyes-free interaction makes complex user interaction possible without interrupting the workflow.
- Auditory or haptic feedback is given to support eyes-free interaction even more.
- The general approach allows application of the design to a wide array of new and existing applications.
- The ubiquity of smart phones and tablets and their usability for other tasks make this approach very cost-effective.
- As the system uses its own mobile screen it can replace large parts of the application's graphical user interface (GUI) up to the complete GUI in some cases.
- Support for multiple users and the system scales well to the number of users.

Functionality of the host application is abstracted to a point where an interaction is basically a binary action (like a button press) or a value manipulation (such as slider controls). While this abstraction might sound like difficult to achieve task or a constraint of this method, it it actually possible to map all common functions to these two methods. This design is universally usable and avoids the need to design new mobile applications for new or updated host applications. It is only necessary to create a reasonable menu structure and connect the existing functionality to it. This speeds up the development process by a big amount. It also has the benefit to present the user with a uniform user interface, reducing adaption time to a new application.

RELATED WORK

Remote Interaction Using Mobile Phones

The use of mobile or smart phones as input device has been an active area of research. Ballagas et al. [3] presented two interaction techniques for camera-equipped phones. One uses the phone as a replacement for an optical mouse, with the physical x-y-translation of the phone mapped to a traditional cursor on the screen. At the time, the method incurred a latency of about 200 ms, too much for a practical applications; even today, high latency seems to be an issue for such approaches. This approach also does not increase the number of functions that can be mapped to a device. The second technique discussed in the same paper is a pure selection method, where the phone detects at which object on the screen it is pointed by tracking markers displayed on the screen. These markers only appear for brief moments while the phone's camera is active to reduce display clutter. Both approaches only employ camera phones as direct mouse replacements, missing the opportunity to further improve the available interaction.

Marking Menus

Kurtenbach et al. [7, 8] proposed and evaluated *Marking Menus* as an improved version of radial menus. The main difference between Marking Menus and transitional radial menus is the absence of a completely bounded target area for each menu item in the former. Radial menus simply arrange their items in a circular pattern around a center point in one or more "rings." An item is selected when the selection cursor is within the bounds of the item, and selection is usually affirmed by a button press on the input device (or any other available confirmation gesture). If the selected menu item has subitems, this will cause a new radial menu to pop up at or near the current cursor position. Marking Menus only have a single ring of items, and their respective target areas expand infinitely outwards from the center of the menu in a wedge-like shape. Holding the cursor still in the selection area of an item with sub-items will cause another Marking Menu to pop up showing the sub-items, and a confirmation event (button press, or, in the original example, lifting the pen from the display surface) will select the current item. The main benefit of this menu layout and selection mechanism is eyes-free item selection, which was at the time proposed to address the high latency of pen-based direct interaction displays on then-current workstations. Using Marking Menus, users could either put the pen down on the screen, wait for the menu to appear, and select items and sub-items by drawing a stroke from the center into the (wedge-shaped) selection areas, while experienced users could just draw the chain of strokes used to reach a certain item without even waiting for the menu to pop up. (Kurtenbach called this "Expert Mode.") Since the selection areas are theoretically infinitely large, user accuracy is very high and Marking Menus are easy to use. These assertions are backed up by Fitts' Law [5] and Steering Law [1]. While the original high-latency problem Marking Menus were designed to address no longer matters, their effectiveness still does, and Marking Menus have been incorporated into many modern applications.

Pook et al [9] proposed *Control Menus*, a general improvement that also applies to Marking Menus: instead of using menus only for selections, the continued motion of an input device after an item has been selected is used to change a continuous value associated with that item. They use an example of a "zoom" menu item, where any continued input device motion after selecting that item directly alters the current zoom level. This method could also be applied to related continuous values by using both display axes simultaneously.

Input Device Taxonomy

To design a general input method suitable for as many tasks as possible, it is crucial to understand the properties defining input devices. There are many input device taxonomies, but in this context we need to focus on the properties of tasks to be performed using input devices, not the inherent properties of special devices, e. g., mounted vs. free-standing. Foley et al. [6] presented a taxonomy of six different tasks that input devices need to provide. While this work is more than 20 years old, it still applies today, as confirmed, for example, by Ballagas et al. [2]. The six tasks identified by Foley et al. are:

1. **Position:**
   Set the absolute or relative position of an object in 2D or 3D space.

2. **Orient:**
   Set the absolute or relative orientation of an object in 2D or 3D space.

3. **Select:**
   Select an item out of a list of several items.

4. **Ink:**
   Define a path consisting of one or more positions and orientations. The name Ink refers to the visual line represented by a path. Ballagas et al. [2] referred to this as "Path."

5. **Quantify:**
   Select a number from a continuous or discrete set.

6. **Text:**
   Enter arbitrary sequences of characters.

This list of essential tasks can be reduced further: text input can be achieved by selection of different characters. Ink is basically a set of positioning and orientation tasks. For the back-end of menu design, position and orientation of a device can be encoded as quantification as well. It is important to note that this simplification does not hold true for the user interface design, where position and orientation are perceived by the user. With this in mind the menu needs to be designed to support at least selection and quantification, while providing multiple means for that in order to support related tasks like positioning or orientation.

Buxton [4] proposed an alternative taxonomy focusing more on the actual usage of the device, e. g., hand-held vs. mounted. But he made one important distinction considering quantification in general: Position vs. Motion. This is taken into account in our approach which makes it possible to use both schemas for quantification.

DESIGN

Our approach uses touch screens for interaction feedback via radial menus similar to Marking Menus, described above in Related Work. Kurtenbach categorizes the usage modes of Marking Menus as Novice Mode (waiting for menus to pop up) and Expert Mode (completing interaction before or while a menu pops up). In the remainder of this paper, we refer to Expert Mode as eyes-free mode, to emphasize the fact that it does not require shifting attention away from an environment's main display.

A radial menu is presented to the user when he/she touches the screen (see Fig. 1). Its design is similar to the original Marking Menu. The menu is centered around the initial finger position, and menu items are selected as the finger enters their selection area. If the selected item has sub-items, a submenu will pop up on selection (see Fig. 2).

To reduce display clutter in deep submenu hierarchies, where submenus are drawn on top of their respective parents, only the currently active submenu is drawn fully opaque, while parent submenus are drawn with increasing levels of transparency, i. e., the root menu is most transparent. Drawing the entire menu hierarchy in this way helps users to see their current position in the menu hierarchy and the direction of the most recent stroke, should they lose their place during eyes-free interaction.

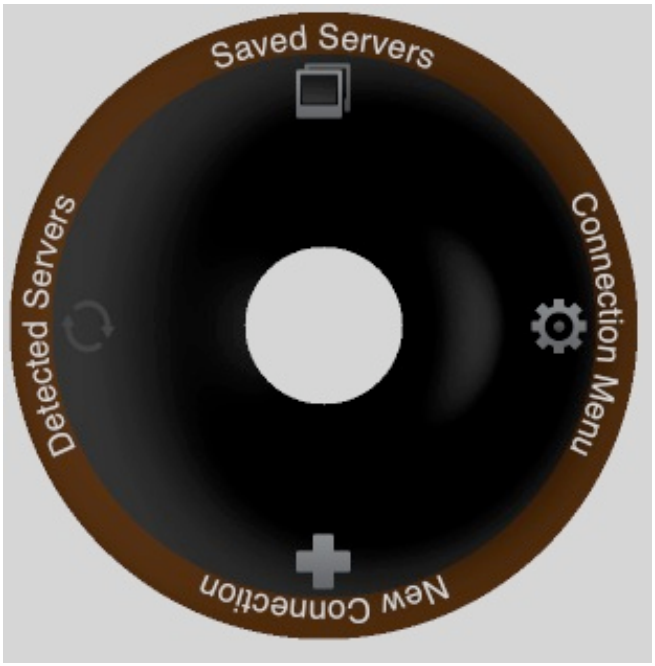

Figure 1. The general design of the menu, in this example for four menu items, all with a descriptive name and an icon.

Our menu design also supports value selection over one or multiple dimensions, employing multi-touch. Our design allows two modes of operation: it can work like a regular slider control, reporting the absolute position of the touch(es) to the application or only the changes in position are reported to the application. This distinction represents the difference of position vs. motion as proposed by Buxton [4] (see section).

MENU DESIGN AND PROTOTYPING

The most important part of this design is the fact that the mobile application receives the information about the menu design from the host application. Therefore only one application on the mobile device is needed to control multiple host applications. There is no need to develop and deploy a new version of the mobile application when the host program changes. This is an even bigger advantage, potentially saving time and money. In our architecture the menu layout is controlled via configuration files that are independent of the program implementation, so changes to the menu on the mobile device can be done without the need to recompile the actual program, allowing for faster prototyping. Output of accelerometers is provided as a simple value manipulation, allowing one to make use of these easily. Formally the smart device is visible to the host application as two (mathematical) functions:

$$b(N) \rightarrow \{0, 1\} \tag{1}$$

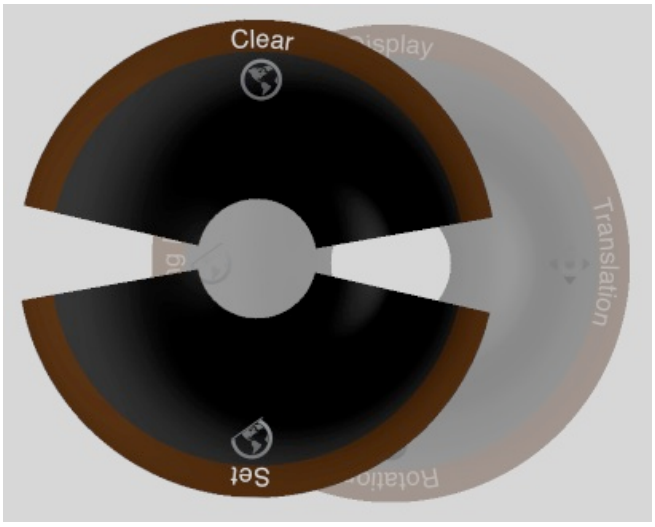

Figure 2. The menu with a submenu opened up after the user moved a finger to the left. The submenu is on top of the half-transparent parent menu.

$$v(N) \rightarrow [-1, 1] \tag{2}$$

Less formally, there are buttons with either the state of 0 (released) or 1 (pressed). These states can be set, reset or toggled by selection of a menu option. This is encoded in function *b*. *v* are valuators, that contain normalized values (normalized to the range [-1, 1]). These valuators can be changed using the slider methods described above, or by using the device's sensors (e.g. accelerometers). The host application needs to map these valuators and buttons to its internal functions, a procedure that has to be done only once for each available function. The layout of the menu (and therefore of the mapped functions) can then be changed easily without having to change any internal implementations of the host application. Also the mapping allows to easily transform and/or combine incoming values for the host application should the need arise. This serves as a kind of model kit for interaction designers to work with and come up with a usable user interface on the mobile device quickly and easily.

CONCLUSIONS

We have presented an eyes-free interaction method using consumer level smart phones and tablets. Being eyes-free users can interact without interrupting their workflow to look at the input device. With its simple consistent design it can replace traditional menus and even whole dialog boxes. Our method is application-independent and can be used in other environments (e.g., Desktop) as well. Applications do not have to be altered to make use of this approach, but the prototype can be customized to include application-specific behavior.

REFERENCES

1. Accot, J., and Zhai, S. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM (1997), 295–302.
2. Ballagas, R., Borchers, J., Rohs, M., and Sheridan, J. G. The Smart Phone: A Ubiquitous Input Device. *IEEE Pervasive Computing 5*, 1 (Jan. 2006), 70–77.
3. Ballagas, R., Rohs, M., and Sheridan, J. Sweep and Point and Shoot: Phonecam-based Interactions for Large Public Displays. In *CHI'05 extended abstracts on Human factors in computing systems*, ACM (2005), 1200–1203.
4. Buxton, W. Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput. Graph. 17*, 1 (Jan. 1983), 31–37.
5. Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 6 (1954), 381–391.
6. Foley, J., Wallace, V., and Chan, P. Human factors of Computer Graphics Interaction Techniques. *IEEE Computer Graphics and Applications 4*, 11 (1984), 13–48.
7. Kurtenbach, G., and Buxton, W. The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, ACM (New York, NY, USA, 1993), 482–487.
8. Kurtenbach, G., and Buxton, W. User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, CHI '94, ACM (New York, NY, USA, 1994), 258–264.
9. Pook, S., Lecolinet, E., Vaysseix, G., Ura, E. C., and Bp, M. Control Menus : Execution and Control in a Single Interactor. *CHI '00 extended abstracts on Human factors in computing systems*, April (2000), 263–264.