ToolStone: Effective Use of the Physical Manipulation Vocabularies of Input Devices

Jun Rekimoto and Eduardo Sciammarella
Interaction Laboratory
Sony Computer Science Laboratories, Inc.
3-14-13 Higashigotanda
Shinagawa-ku, Tokyo 141-0022 JAPAN
Phone: +81 3 5448 4380
Fax +81 3 5448 4273
rekimoto@acm.org, http://www.csl.sony.co.jp/person/rekimoto.html

ABSTRACT

The ToolStone is a cordless, multiple degree-of-freedom (MDOF) input device that senses physical manipulation of itself, such as rotating, flipping, or tilting. As an input device for the non-dominant hand when a bimanual interface is used, the ToolStone provides several interaction techniques including a toolpalette selector, and MDOF interactors such as zooming, 3D rotation, and virtual camera control. In this paper, we discuss the design principles of input devices that effectively use a human's physical manipulation skills, and describe the system architecture and applications of the ToolStone input device.

KEYWORDS: Interaction techniques, input devices, physical user interfaces, multiple function inputs, multiple-degree-of-freedom input, two-handed input

INTRODUCTION

Although the mouse is the most successful input device in the history of computer interfaces, its limits are becoming frustrating as the complexity of software increases. The mouse is a generic input device, so a user must first be able to see a command (such as a menu item or tool button) on a screen, and then select it before the command is actually put into effect. For example, when a user wishes to draw a circle in a drawing editor, the user would open a toolpalette containing a circle tool, select the tool, then start drawing. These command objects are spatially deployed around the application (e.g., on tool bars or scroll bars), or appear according to the user's operations (e.g., pop-up menus or toolpalettes). Selection of these commands requires both physical (manipulation of an input device) and visual (recognizing a tool button and a cursor on a screen) efforts.

While this operating style has been effective for relatively simple software applications, an increasing number of functions, makes it more cumbersome to specify appropriate functions (menus, tool palettes, or property sheets). Many modern (and feature rich) applications use several toolpalettes and tool bars, and these take up screen space leaving less space for actual use. It has become impossible to lay out all available tools on a screen, so users have to frequently open and close tool palettes according to the task. This trend is forcing us to use bigger computer displays, but moving the mouse cursor between tool areas and application areas (such as a drawing canvas) becomes more time-consuming as the screen size increases.

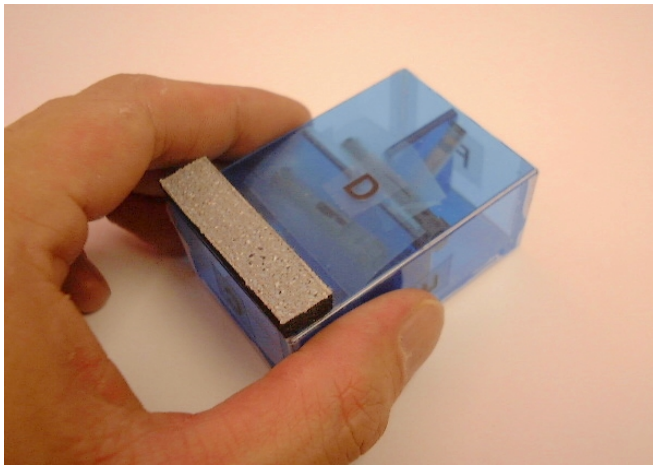

Figure 1: The ToolStone: a cordless semi-6DOF input device. Coils embedded in the device are used to measure position, orientation, tilt angle, and contacting face when it is placed on a tablet surface.

In contrast, physical tools allow effective use of a human's rich manipulation skills, and a single physical tool can often be used in many different ways. To illustrate this difference, Gershenfled compared the mouse with the violin bow [12]; while the mouse only provides a limited set of manipulation vocabularies (such as clicking or dragging), the violin bow has hundreds of ways in which it can be used. Trained violin players can easily change between tones (i.e., interaction modes) quite rapidly, and this selection relies heavily

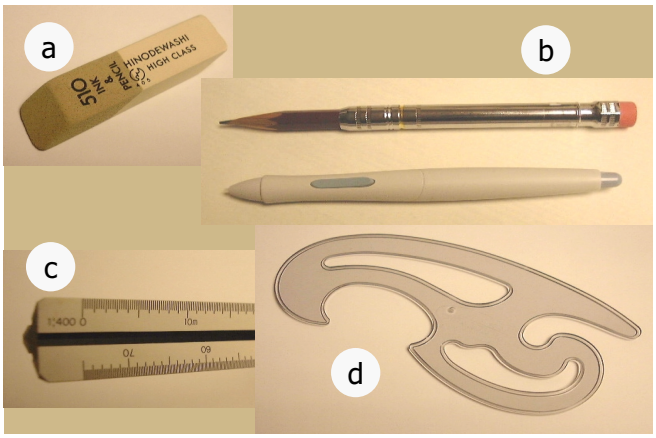

Figure 2: Multifunction physical tools: (a) a two-way rubber eraser, (b) a pencil with an eraser at one end and, its digital adaptation (the WACOM stylus), (c) a scale with six different divisions, and (d) a French curve with which a user can draw several different curves.

on motor skills: visual attention directed to the tool is not required.

Although directly comparing the mouse and the bow might be extreme, there are many daily tools that also provide multiple function through a single physical object. Figure 2 shows some examples of these. One thing these examples have in common is that we change the way we holding them to perform different functions. With a pencil that has a rubber eraser at one end, for example, we can easily change from a draw mode to eraser mode by simply reversing our grip.

In this paper, we explore the idea of expanding the functionalities of a single input device, and enabling users to select functions by changing the way they hold the device. Although this technique is related to multiple-degree-of-freedom (MDOF) input devices, we are also interested in developing interaction techniques that are not limited to the manipulation of 3D objects. We refer to such a physically enriched input style as a rich-action input.

In this paper, we discuss the design principles for rich-action input devices, then describe our ToolStone input device (Figure 1). The ToolStone is a cordless, MDOF interaction device that is designed to be easily rotated and flipped to activate several different functions.

RELATED WORK

Much research has been aimed at enhancing the "richness" of input devices. Embodied User Interfaces [13] attach several sensors to increase the usability of PDA. Tagged Handles allow a user to attach different handles to a rotational rod [17]. Users can differentiate between the functions of tag handles both visually and physically. The Cubic Mouse is a 6DOF input device with pushbuttons and movable penetrating shafts [11]. These shafts are used to provide additional operation modes; such as changing a cross-sectional plane of a 3D object.

There are also several examples of using the movements of input devices. Tilting user interfaces [19] use the tilt of portable devices as input. For example, a tilt sensor embedded in a hand-held computer can be used to scroll for menu selection or map scrolling. Embodied User Interfaces [13, 9] and Rock'n'Scroll [4] also use tilt interfaces.

The Rockin'Mouse is a mouse with a tilt sensor that can be used to manipulate a 3D object [3]. Kuroki and Kawai proposed the use of tilt information for pen interfaces [15]. They observed that people hold three physical tools (a pencil, a knife, and a syringe) differently, and they built a pen interface that allows a user to select different functions by changing its tilt on a tablet based on this observation.

We have also explored several interaction techniques that can be used when motion sensing becomes available in hand-held devices [21, 2]. For example, when a user places a PDA near an object displayed on a digital whiteboard, the PDA becomes a toolpalette for that object and the user can 'click through' a command by tapping on the PDA. Likewise, sweeping the surface of a digital table with a PDA enables data transfer between them just as we sweep breadcrumbs from a table into a dustpan.

Some researchers have also proposed associating multiple functions with a single object. PadMouse is a mouse with a touch-pad instead of a button. A user can make a finger gesture on a pad to select different functions. Fitzmaurice described the concept of flipbricks as part of his graspable user interfaces [10]. On each face of a flipbrick device, different commands, such as "cut" or "copy" are associated and users can activate one of them by flipping the device. Want et al. proposed an augmented photo-cube, a block with six wireless tags attached to its faces [23]. Up to six different digital contents can be associated with these tags, and can be retrieved by touching a face with a tag reader.

Our ToolStone uses multiple faces for different functions, and further increases the number of selectable functionalities by combining other manipulation vocabularies such as rotation or tilting. The ToolStone also uses several interaction techniques based on the physical movement of the ToolStone itself during operations.

DESIGN PRINCIPLES FOR RICH-ACTION INPUT DEVICES

In designing input devices that allow a user to select an appropriate function through physical actions, the following principles are important.

The device's state should be perceived through touch

Well-designed physical devices often reveal their operation mode without relying on visual information. While the user is concentrating on a task, the physical device's feel implicitly shows its state. Figure 3 shows two examples. The first example (a) is the three-state button of a video camcorder. While this camcorder is held by hand, the user's thumb always touches this button so the user can perceive the camcorder's state (camera-mode, playback-mode, and off) through the tactile impression. In contrast, a user of the camera in the second example (b) has to look to see the setting of a dial

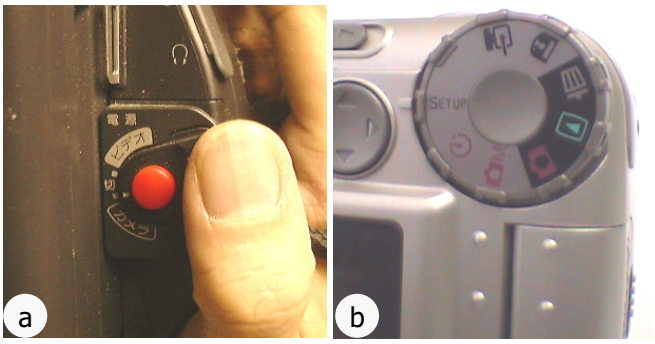

Figure 3: Tactile impressions reveal the state: (a) A camcorder switch with three states. A user can physically perceive the current state from the thumb position on the switch during operation. (b) A dial of a digital camera. In this example, the physical shape does not change so visual labels (hence, visual attention) are needed to know the current state.

because its shape does not indicate the selected mode. Labels on the dial are necessary, and a user must read these.

If we assign multiple functions to one input device, its state should be perceivable from tactile impressions; as well as through visual feedback, so that determining the currently selected state does not distract a user's visual attention.

This is one reason why a physical dial, such as the rotating dial in the WACOM 4D mouse, is not an effective way to select a function. With such a device, a user needs to find out the current state through visual feedback which may distract the user's visual attention. A second reason is a dial's sequential feature; instead of selecting a function in one operation, a user has to change the states one at at time by rotating the dial until the desired function becomes available.

The interaction space should be easily understandable

Although it is technically possible to implement a number of functionalities in a single input device, it is useless unless users can find them. Thus, visual appearance of the device is still important, in that it can help users visually recognize available functions at a glance. For example, a camcorder user (Figure 3(a)) would first understand the function of the switch from its visual appearance, and would then gradually learn to manipulate it by touch.

THE TOOLSTONE

To explore the benefits of input devices that support richer physical manipulations, we built the ToolStone device (Figure 1). The ToolStone is a cordless, rectangular object that is designed mainly as an input device for the user's non-dominant hand with bimanual interfaces (Figure 4). While the dominant hand manipulates a pointing device such as a mouse or a stylus, a ToolStone held by the non-dominant hand is used to select appropriate functions, or to provide more flexibility in operations.

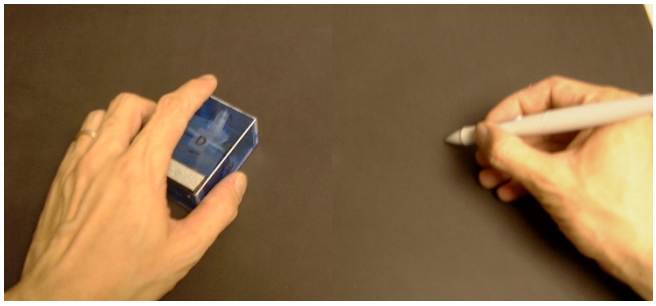

Figure 4: Bimanual interaction with the ToolStone.

The ToolStone is a semi-6DOF input device. When placed on a tablet, its x-y positions and orientation are measured. The tablet also detects which face of the ToolStone is touching the tablet surface. When one of the edges is touching the tablet, the tilt angle can also be measured.

A small projection (a bar) is attached to the lower edge of one face. By feeling this projection, a user can perceive the device's orientation and face direction without visual/audio feedback (Figure 5).

INTERACTION TECHNIQUES

Although the ToolStone is not a complete 6DOF input device (at least one face or edge has to be touching the tablet during operation), several new interaction techniques can be realized.

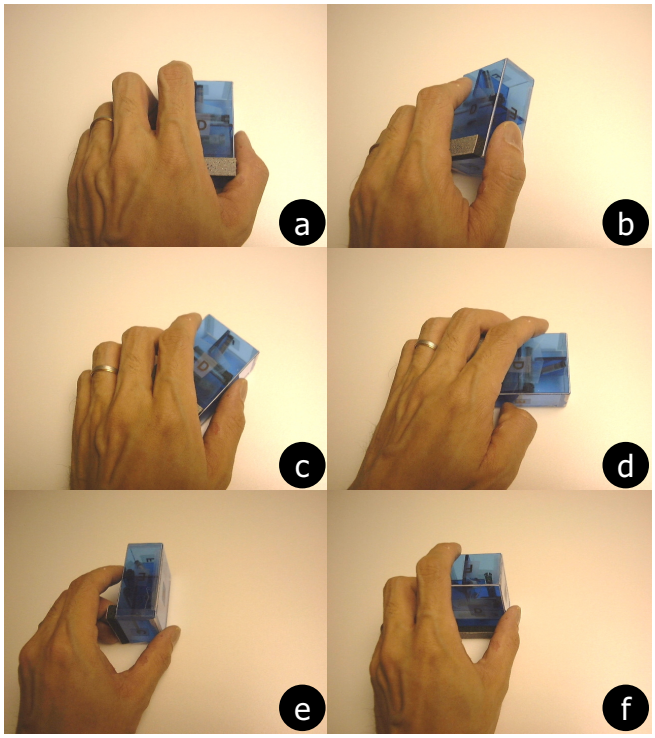

Figure 5: Several possible ways of holding the ToolStone: (a) Normal mode (Note: a projection attached near the lower edge of the upper face can be felt by the hand). (b) Tilting while one edge is contacting the tablet (c, d) Rotating, and (e, f) Flipping to select other faces.

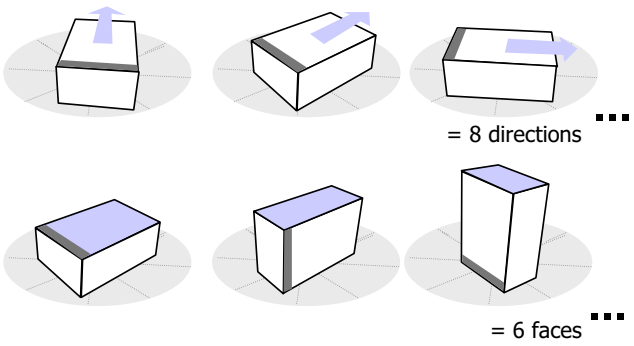

Figure 6: Selecting multiple functions by rotating and flipping the ToolStone: The combination of eight directions and six faces allows a user to quickly select 48 different functions (e.g., toolpalettes) with a single physical action.

This section briefly describes the typical use of the ToolStone.

Tool selection

When used as a non-dominant hand device for bimanual interfaces, the ToolStone can be used as a tool selector for the dominant-hand input device. For example, eight different toolpalettes, each with several different command items, can be assigned to eight directions separated by 45 degrees, and a user can quickly select an appropriate palette by rotating the ToolStone. Furthermore, the user can switch to a different set of tools by flipping the ToolStone to select another face. If a set of eight toolpalettes are attached to each face, and the six faces have different sets, 48 different toolpalettes can be selected by through a single physical action (Figure 6). This would meet the requirements of most real-world applications.

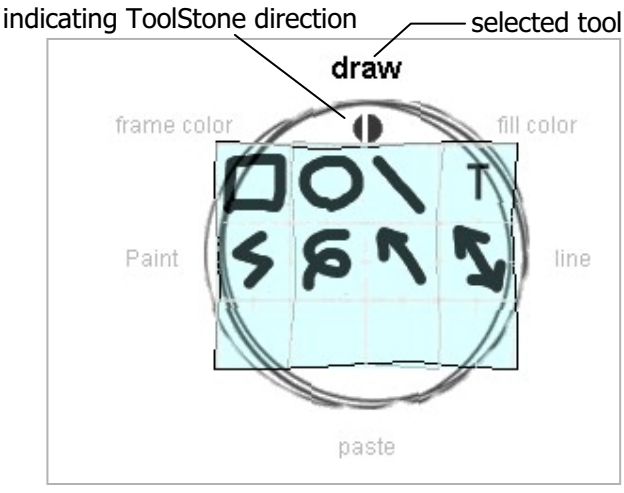

Figure 7: Example of a selected toolpalette: A dial and labels around the tool palette indicate available functionalities attached to the same face. The currently selected one is shown in bold. The selected toolpalette acts as a ToolGlass sheet.

Figure 8: A ToolStone device with labels on each face. A (novice) user would be able to visually inspect available commands by physically turning the device.

This feature is particularly suitable for selecting toolpalettes in ToolGlass or MagicLense interfaces [6, 5]. In the original ToolGlass design, the non-dominant hand is only used to control the location of a ToolGlass sheet. With the ToolStone, it can also be used to switch between several toolpalettes (ToolGlass sheets) with a quick physical action. Since only one toolpalette appears on the screen at a time, the screen would not be cluttered by a number of floating palettes, as is often the case with today's application software.

The form-factor of the ToolStone is designed to enable comfortable manipulation. The width, height, and depth of the ToolStone are all different; combined with the attached projection, this allows the user to easily distinguish the physical state.

In addition, it is useful to add labels to the ToolStone faces, so that (novice) users can visually inspect the available functionalities by physically turning the device in their hands (Figure 8).

An interesting feature of the ToolStone is that we can organize the command space physically. For example, when we assign related functionalities (such as tools for picture element creation and tools for giving a color to an element) to adjacent positions (i.e., adjacent angles), the ToolStone's physical manipulation distance (the time required for switching between two functions) would also represent the logical distance between tools. Currently we assign a color selection tool and picture creation tool to adjacent angles of the same face, and file manipulation commands to another face. After creating a picture element, a user can slightly rotate the ToolStone (45 degrees) to get the color selection tool. As a user repeatedly performs this sequence, we expect that it would become a chunk of physical operations.

MDOF interaction techniques

When one tool is selected, the ToolStone's x-y positions are still available for other manipulation. We can use these values

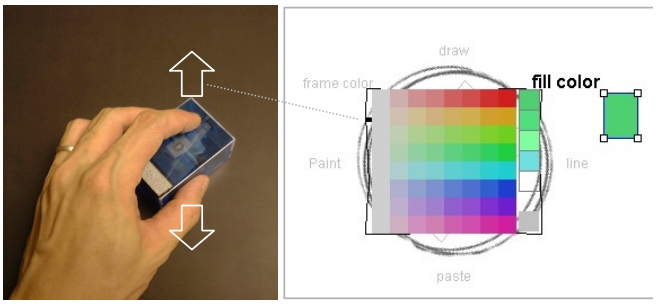

Figure 9: A color selection tool example: ToolStone's vertical motion controls the brightness parameter of the color space, while two other parameters (hue and saturation) are mapped according to the x and y axes of a 2D palette. A user can dynamically navigate though the color space before selecting a color instance. Note that the direction of the ToolStone is used to select the color selection tool.

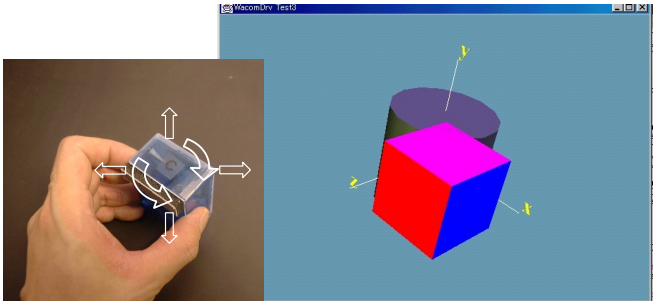

Figure 10: MDOF movement of the ToolStone can be mapped for 3D object control.

to control the position of the selected toolpalette in ToolGlass-type interfaces.

The other possibility is to use them for controlling parameters during toolpalette operations. For example, for a color selection toolpalette, the forward/backward movement of the ToolStone can be used to control the brightness parameter (Figure 9). Since color space is a 3D space (e.g., hue-saturation-brightness), color selection requires control of three parameters. Existing color selection tools often force unintuitive operations because of the bad mappings between the 3D color space and the 2D toolpalette space. Our solution allows a user to simultaneously control the third parameter (e.g., brightness) by moving the ToolStone device, while the dominant-hand pointing device selects a point on a toolpalette.

It should also be possible to apply this idea for various kinds of interactions that require more than 2D parameter control. For example, a World within a World interface [8] for exploring up to a 4D information space can be implemented as a 3D graph, and remaining 2D parameters can be manipulated by forward/backward and sideways movement of the ToolStone.

MDOF control

In addition to the MDOF interaction techniques described above, it is possible to simultaneously control parameters of more than two degrees of freedom.

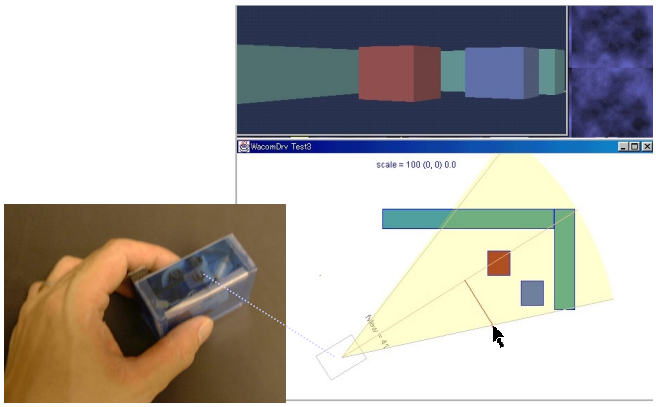

Figure 11: A user is manipulating a virtual camera of a 3D world. While the non-dominant hand is used to control the camera's position and orientation, the user can also change the field of view by dragging a viewing area (projected as a filled arc) with the dominant-hand's pointing device. Note that the pointing device is also used to change the viewing angle of the camera.

For example, one face of the ToolStone can be assigned to zooming and panning of the workspace. Without moving the cursor to the scrollbars at the edges of a window, a user can select a zooming tool by flipping the ToolStone. The ToolStone's forward/backward and sideways motions are mapped to scrolling, while its rotation controls scaling. For example, rotating the ToolStone clockwise can be mapped to increasing the scale (i.e., zooming in).

Another example is 3D rotation of an object. When a user selects an object on a screen and holds it with the dominant-hand's pointing device, the ToolStone becomes a rotation tool. For example, the horizontal and vertical motions of the ToolStone control the direction of the rotation axis, and its rotation controls the angle of object rotation (Figure 10).

Figure 11 shows another example of combining tool selection and MDOF control. When a user flips the ToolStone to select a virtual camera tool in this 3D scene-building application, a 3D view window appears and the user can control the viewpoint of the camera by manipulating the ToolStone as a physical camera on a floor plan. During this operation, the dominant-hand pointing device can also be used to alter interaction parameters. For example, the field of view of the camera can be directly manipulated by dragging an edge of a view frustum that is projected onto a floor plan.

SYSTEM IMPLEMENTATION

Sensor architecture

To enable the interactions described in the previous sections, we needed a means of sensing ToolStone's orientation, which of it's faces is in contact with the tablet surface, and its position. The ability to measure these parameters "untethered", freeing the user from the bother of a wire during operations, was also desirable. Since most MDOF input devices (such as the Poluhemus isotrak [18]) are tethered devices, we decided

to design our own sensor architecture.

Our first implementation was based on visual sensing. We attached six different visual patterns to the ToolStone faces and placed it on a semi-transparent acrylic board that acted as a tablet surface. A camera below the acrylic board was used to determine the position and orientation of the ToolStone, and to detect the contacting face. As a prototype, this architecture worked reasonably well, but the tablet was too thick. We thus looked for an alternative solution based on the widely used electro-magnetic pen tablet.

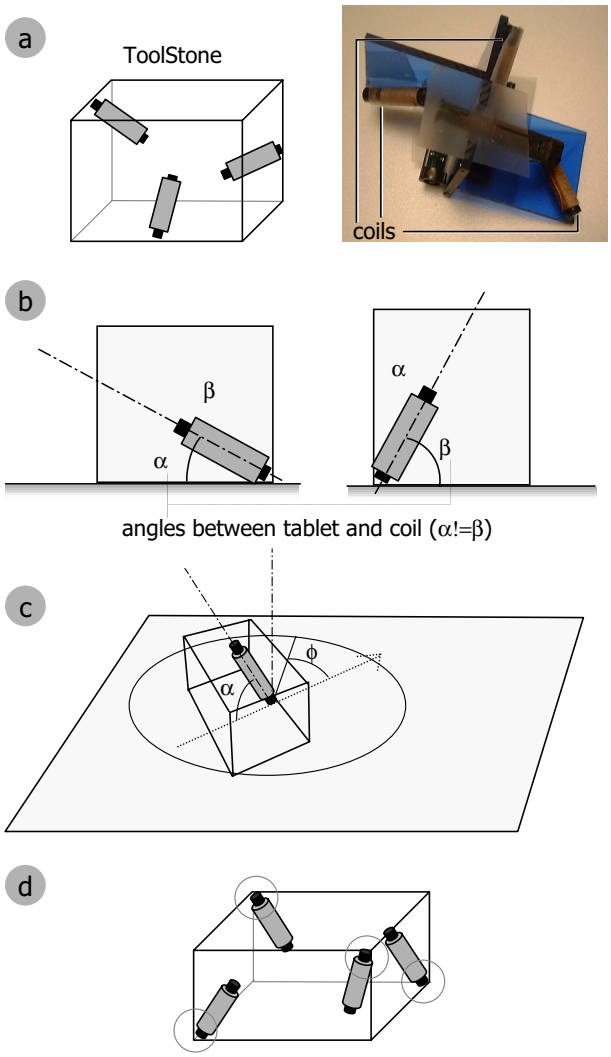

Figure 12: Detection of the touching face and orientation: (a) Inside the ToolStone: Three WACOM coils are embedded, and only one of them will be close enough to the tablet surface when the ToolStone is placed on the tablet. (b) When a coil touches the tablet, it can be identified by its unique resonance value. Two faces that share the same coil can be distinguished by comparing the tilt values (α and β). (c) Once the touching face is known, the orientation of the ToolStone can be determined from the orientation angle of the coil (ϕ). (d) An alternative sensor configuration with coils at the four corners of the device. Two of these coils are in contact with the surface when one face is placed on the tablet.

We used the WACOM tablet(WACOM UD-II series) [7] as our next platform and developed a ToolStone with a three-coils architecture (Figure 12). We embedded three coils, taken from WACOM styluses, at three different edges of the ToolStone. The WACOM tablet emits magneto-electric signals to the nearby area, and a coil with a specific resonance parameter responds to this signal. By analyzing this response pattern, we can measure the coil's position on a tablet, as well as its angle and orientation.

When one of the ToolStone surfaces touches the tablet, only one coil is in contact with the tablet (Figure 12). Although this coil is shared by two faces, the system determines which face it is by measuring the angle of the coil. The orientation of the ToolStone can also be calculated from the coil orientation when contacting face is known.

Each of the three coils can be identified through its unique resonance parameter. The original WACOM stylus consists of a coil and a small ferrite core that is combined with a small spring. This mechanism is used to measure the pen pressure. When a user changes pen pressure, the system measures the resonance parameter of the pen which will vary according to the distance between the coil and the ferrite core. To use this value to determine which coil touches the tablet surface, we attached small ferrite cores to the three coils, each at a slightly different distance. The coils can thus be detected in the same way as three pens with different pen pressures.

Combining these features made it possible to determine the touching surface of a ToolStone, as well as its position and orientation in relation to the tablet, without using wires or batteries. In our prototype design, the ToolStone is 2.5 $\times$ 4 $\times$ 5 cm, and it weighs 22 g. This is much smaller and lighter than a conventional mouse. The weight and form-factor make it easy to manipulate in a user's hand.

Since only one coil (out of three) needs to be sensed at one time, a tablet that supports simultaneous sensing of two objects can be used as a bimanual manipulation tablet (most commercially available tablets can simultaneously sense only two objects).

We are also planning to attach a button to one face of the ToolStone, so that it can also operate as a normal mouse.

Software architecture

For application programmers, we have developed a ToolStone device driver interface of 'raw' tablet driver [16]. This layer hides the internal recognition algorithm, and provides an event-driven interface to applications. For example, a ToolStone-aware application is programmed to receive a "stone" event, as well as mouse events. The stone event contains information concerning the ToolStone status, including the currently selected face, position, and orientation.

This driver interface was written in C on Windows 98, and all example applications described in the previous sections were written in Java. These Java applications communicate

to the ToolStone driver layer through the Java Native Interface (JNI). Applications that support 3D object manipulation were built with Java 3D.

PROTOTYPE TRIAL

To date, we have implemented a simple drawing tool and interaction techniques based on the ToolStone. Five pilot users (all were expert with GUI tools, but not familiar with two-handed interfaces) have tried the system after a minimal demonstration. Although a formal user study is still being planned, we obtained some interesting feedback during this trial.

All users instantly understood the ToolStone concept and could easily select different tools. Some users preferred to keep the same face down and rotate the ToolStone, rather than to flip it, mainly because these was less physical motion and sound generated than when it was flipped.

To provide visual cues, the current implementation used labels around a currently used tool to indicate other available functionalities (Figure 7), but this information was limited to the functions that belonged to the same face. Many user required similar labels for other faces.

Some users told us that they felt there was a strong relationship between the spatial manipulation and the tool space. One user compared the ToolStone to an analog clock, and explained his image of all the tools being assigned on a dial of a clock. Another user mentioned that he could easily remember the assignment of the functions when he imagined he was manipulating a small doll instead of a rectangular shape.

Some users explained that they could remember a sequence of hand actions in the same way we remember word spellings when touch-typing. We observed that one user, who was quite accustomed to the prototype application, had difficulty when he tried to recall the assignment of tools without actually manipulating the ToolStone. In our daily lives, we often use physical skills that we can apply but cannot explain in words. Whether the ToolStone requires the use of similar physical skills is an interesting question.

CONCLUSION AND FUTURE DIRECTIONS

A rich-action input device allows users to interact with computer functionalities by physically changing the way they hold the input device. The ToolStone, a cordless multiple-degree-of-freedom (MDOF) device is such an input device. A ToolStone's unique sensor architecture allows the system to sense physical manipulation of the device itself; for example, rotating, flipping, or tilting. As an input device for the non-dominant hand with bimanual interfaces, the ToolStone can be used, for example, as a tool selector, for MDOF interactions such as zooming, 3D rotation, or virtual camera control.

ToolStone is still at an early stage of development, though, and there are several directions for further study. Our research topics for the immediate future are explained in the following.

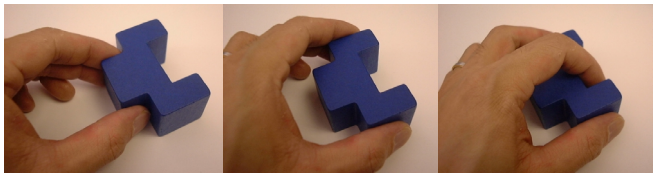

Figure 13: An object with several different ways of holding

Figure 14: Several design variations of the ToolStone shapes.

Evaluation of other physical shapes

Our initial prototype was rectangular but other shapes are also worth considering. One possibility is a polygonal (e.g., hexagonal) pyramid with its top cut off. With this shape, a user can select a face with less physical motion than is required with the present shape. We may also add distinctive physical textures such as small holes or grooves to every surface, or round edges or faces to make tilting motions easier.

As an alternative to moving the device, it may also be possible to select functions by detecting the way the user touches the device. By extending the idea of the touch-sensitive mouse [14], we can attach touch sensors to the faces of the input device. Thus, users may be able to switch between operating modes by changing their grip on the device. For example, the physical object shown in Figure 13 can be held in several different ways.

Also, other shapes may be more esthetically appealing than a simple rectangle. Figure 14 shows some shapes that create stronger positive impressions, and we expect that future

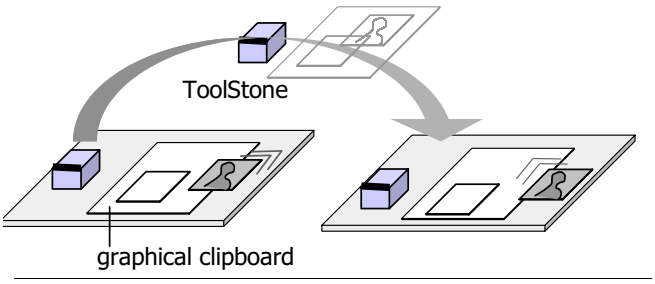

Figure 15: The ToolStone is used to transfer data from one computer to another: A user can carry the contents of the graphical clipboard with the ToolStone.

computer applications will be symbolized by their own unique 'stone' shapes.

Multiple ToolStones

For highly complicated tools, such as a high-end graphic tool, it is also possible to use more than one ToolStone device. Each stone object represents a category of operations, such as 3D modeling tools or photo retouching tools. A user can switch the operating mode by physically exchanging one ToolStone for another. Another idea is to provide a different ToolStone set for different categories of users. A software application may behave differently with different kinds of ToolStone. For example, a ToolStone for children might provide only basic functions, while a ToolStone for adults would also provide more complicated functions.

The ToolStone may also act as a physical information carrier between computers by using techniques similar to Pick-and-Drop [20] or mediaBlocks [22]. In this scenario, a user can copy data from one computer to the ToolStone, and retrieve it when using another computer. For example, a user could display a graphical clipboard panel on one computer then drag an object onto the clipboard (Figure 15). When the ToolStone is removed from the tablet, the clipboard panel is removed with it and the user virtually carries it with the ToolStone. When the user places the ToolStone on a different computer, the same clipboard re-appears and the user can drag-out any of the carried objects. (We assume that only the ID would be stored in the ToolStone and the actual data transfer would be done through the network).

Study of human memory and computer input

Finally, we would also like to study the human memory skills required to deal with computer systems. According to cognitive psychology theory (such as [1]), a human's long-term memory can be classified into two major categories: declarative memory (i.e., knowledge that can be explained by words), and procedural memory (i.e., learned skills). In our daily lives, we rely heavily on the latter so that we can concentrate on tasks requires active use of knowledge. It seems, however, that today's computer systems still relay too much on users' declarative memory, and do not effectively utilize learned skills. Our experience with the ToolStone suggests that input devices would be more effective if they made better use of human motor skills.

ACKNOWLEDGEMENTS

We thank Haruo Oba and the members of the Sony CSL Interaction Laboratory for helping us to explore the ideas described in this paper. We also thank Brygg Ullmer and George Fitzmaurice for introducing us to flipbricks. Finally, we are indebted to Mario Tokoro and Toshi Doi for their continuing support of our research.

REFERENCES

1. Mark H. Ashcraft. *Human Memory and Cognition*. Addison-Wesley Publishing Company, 1994.

2. Yuji Ayatsuka, Nobuyuki Matsushita, and Jun Rekimoto. HyperPalette: a hybrid computing environment for small computing devices. In *CHI 2000 Extended Abstracts*, pages 133–134, 2000.

3. Ravin Balakrishnan, Thomas Baudel, Gordon Kurtenbach, and George Fitzmaurice. The Rockin'Mouse: Integral 3D manipulation on a plane. In *CHI'97 Proceedings*, pages 311–318, 1997.

4. Joel F. Bartlett. Rock'n'Scroll is here to stay. *IEEE Computer Graphics and Applications*, 20(3):40–45, 2000.

5. Eric A. Bier, Maureen C. Stone, Ken Fishkin, William Buxton, and Thomas Baudel. A taxonomy of see-through tools. In *Proceedings of CHI'94*, pages 358–364, 1994.

6. Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony DeRose. Toolglass and Magic Lenses: The see-through interface. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 73–80, August 1993.

7. Wacom Technology Co. Product information. http://www.wacom.com.

8. Steven Feiner and Clifford Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 76–83, Snowbird, Utah, October 1990.

9. Kenneth P. Fishkin, Thomas P. Moran, and Beverly L. Harrison. Embodied User Interfaces: Towards invisible user interfaces. In *Engineering for Human-Computer Interaction: Seventh Working Conference on Engineering for Human-Computer Interaction*, pages 1–18, 1998.

10. George W. Fitzmaurice. Graspable user interfaces. *Ph.D thesis, University of Toronto*, 1996.

11. Bernd Frohlich and John Plate. The Cubic Mouse: A new device for three-dimensional input. In *CHI'2000 Proceedings*, pages 526–531, 2000.

12. Neil A. Gershenfeld. *When things start to think*. Henry Hold & Company Inc., 1999.

13. Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *Conference proceedings on Human factors in computing systems (CHI'99)*, pages 17–24, 1998.

14. Ken Hinckley and Mike Sinclair. Touch-sensing input devices. In *CHI'99 Proceedings*, pages 223–230, 1999.

15. Tsuyoshi Kuroki and Satoru Kawai. An interface technology for pen devices using tilt information. In *Interactive Systems and Software VII*, pages 1–6. Kindai Kagaku Publishing Inc., 1999.

16. LCS/Telegraphics. Wintab: Advanced pointing device management for windowed environments. http://www.pointing.com.

17. Karon E. MacLean, Scott S. Snibbe, and Golan Levin. Tagged Handles: Merging discrete and continuous manual control. In *CHI'2000 Proceedings*, pages 225–232, 2000.

18. Polhemus, Inc., Colchester, Vermont. *3SPACE ISOTRAK User's Manual*, 1987.

19. Jun Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '96)*, pages 167–168, 1996.

20. Jun Rekimoto. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proceedings of UIST'97*, pages 31–39, October 1997.

21. Jun Rekimoto. A multiple-device approach for supporting whiteboard-based interactions. In *Proceedings of ACM CHI'98*, pages 344–351, February 1998.

22. Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. mediaBlocks: Physical containers, transports, and controls for online media. In *SIGGRAPH'98 Proceedings*, pages 379–386, 1998.

23. Roy Want, Kenneth P. Fishkin, Anuj Gujar, and Beverly L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI'99 Proceedings*, pages 370–377, 1999.