Joystick Keyboard: Physically Integrated Typing and Pointing

Wael Aboelsaadat

ABSTRACT
This paper presents the Joystick Keyboard, a multi-finger input device. The Joystick Keyboard has the same shape and layout as a regular keyboard and can perform all the usual functions. In addition, each key has a joystick underneath enabling the user to control two degrees of freedom. With the Joystick Keyboard, the data entry and mouse movement can be both accomplished, eliminating the need for a dedicated pointing device. In this paper, we describe our design in detail as an enabling technology. We then present several multi-finger interaction techniques that exploit the affordance of the device. Finally, we present initial usability testing results, and user reactions to these techniques.

Keywords
Input devices, keyboard, mouse, multi-finger interaction, multi degree-of-freedom input, input gestures.

ACM Classification Keywords
H5.2. User Interfaces.

INTRODUCTION

Multi-finger devices constitute a promising class of input devices, as they enable the user to perform richer and more natural gestures than what is afforded by conventional devices, such as a keyboard or a mouse. However, most of the existing multi-finger input devices were designed primarily for pointing and not typing. Typing still remains the sole preferred mechanism to input significant amount of data.

In this paper, we present the Joystick Keyboard (Figure 1), a new multi-finger input device supporting physically integrated typing and pointing. The Joystick Keyboard takes advantage of mini-joysticks, e.g. CTS252 [3] and EOS10 [5]. These are miniature versions of standard joysticks and have been primarily used in handheld gaming consoles and remote controls. In the Joystick Keyboard, each key is an integrated mini-joystick/switch. The device facilitates a verity of new interaction techniques that exploit the use of multiple fingers. In the following sections, we discuss our design in detail, followed by a look at several interaction techniques and then we present the initial user feedback to our system.

RELATED WORK

Our work relates to two areas of research: integrated input devices and multi-finger input devices.

Integrated Input Devices

Use of a conventional keyboard and a discrete computer device in order to enter typing and pointing information, respectively, into a computer require physical disruption that may significantly reduce a user's productivity. Previous works [1,2] have shown that moving from the keyboard to the mouse takes approximately 40% of the total time it takes to point to an object on a computer screen. A number of data input devices and techniques with combined keyboard/pointer function have been proposed. We call

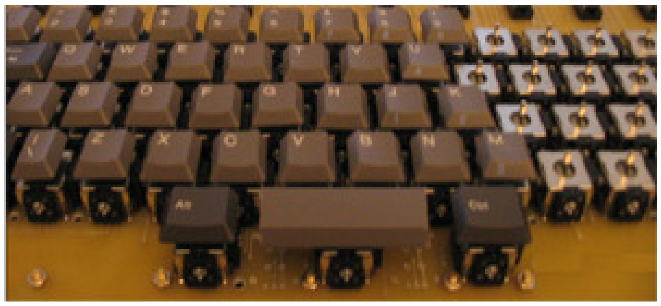


Figure 1. The Joystick Keyboard uses mini-joystick sensors to detect key press and sense multiple degrees of freedom.

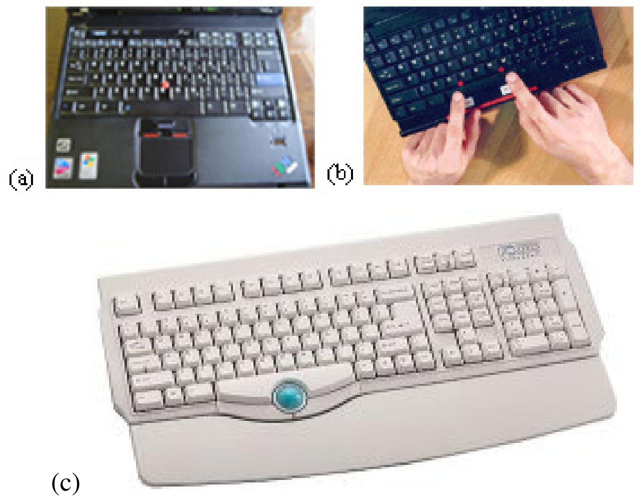


Figure 2. Examples of Integrated Input Devices: (a) Keyboard with integrated Track point and Touchpad, (b) Keyboard with two integrated Track points (c) Keyboard with integrated Trackball.

these integrated input devices because they support pointing and typing. These systems typically have a pointing device (Figure 2), such as a trackball, a touchpad, or a track point [13], installed within the keyboard compartment. These designs reduce but do not eliminate the homing time, as they are based on two or more physically separate hardware components. The user must reposition her fingers whenever moving from one activity to another. As a solution to the homing problem, ThumbSense [12] senses finger contact on the touchpad and uses this information to automatically change operation modes. For example, when a user touches the touchpad, some of the keyboard keys act as mouse buttons so the user can smoothly switch text input and touch-pad operation (mouse) modes without removing their hands from the keyboard's home position. Vision based techniques have also been proposed as an alternative solution. The FingerMouse [9] used a camera to track a pointing finger above a conventional keyboard. The Visual Touchpad [8] used two cameras to track multiple fingers typing on a virtual keyboard. However, these vision based designs lack the tactile feedback associated with typing and pointing.

Multi-finger Input Devices

Multi-finger input devices can be generally classified into surface-based devices and key-based devices (Figure 3). Surface-based devices detect multiple points of contact and are typically used to support gesture-based interactions. Although typing could be accomplished using a virtual keyboard, yet the lack of tactile feedback makes them less attractive in data entry tasks. Examples of surface- based devices are; the DViT sensing technology [14] from Smart Technologies , which uses computer vision techniques to

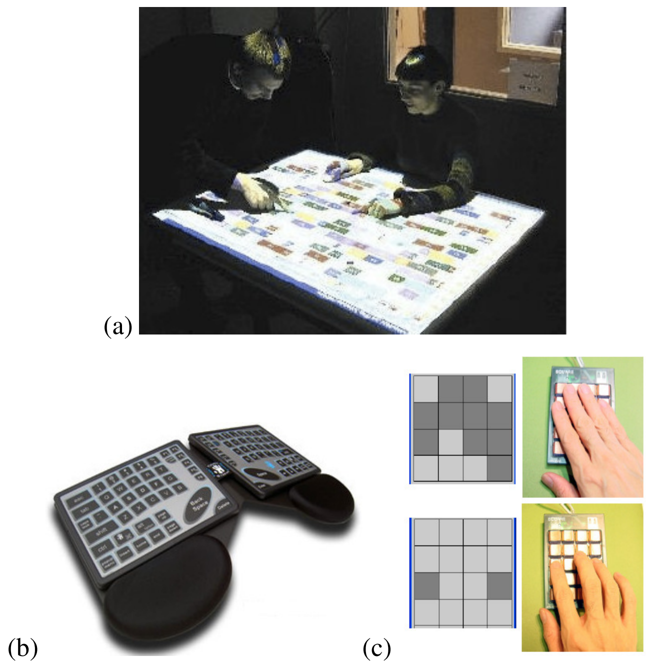


Figure 3. Examples of Multi-finger Input Devices: (a) DiamondTouch from Dietz et al. [4] , (b) FingerBoard from FingerWorks [6] , (c) Presense Keypad from Rekimoto et al. [11].

sense touches. The DiamondTouch table [4], which works by transmitting signals through antennas in the table. These signals are capacitively coupled through the users and chairs to receivers, which identify the parts of the table each user is touching. The Tactex [15] smart fabric technology, which senses pressure by changes in the optical properties of the material. SmartSkin [10], which recognizes multiple hand positions and shapes and calculates the distance between the hand and the surface by using capacitive sensing and a mesh-shaped antenna.

The second category, where our prototype belongs, is the key-based devices. These devices are characterized by the integration of electro/mechanical sensors into a key-shaped physical enclosure, which enable them to easily support typing and pointing. FingerBoard [6] from Finger works is one example of these devices. Although the FingerBoard is not shipping as of date, it appears to use a two-dimensional array of capacitance sensors to obtain an image of the finger tips touching it, enabling a rich set of multi-finger gestures. The PreSense keypad [11], senses finger motion on the device's surface by using multiple touch sensors instead of using a single sensor that covers the entire device. When a user first puts her finger on the keypad, preview information is provided on the screen. The information can be modified according to the finger position on the keypad. Then, the user can actually execute the previewed command by physically pressing one of the buttons. PreSense also detects gestures performed by tracking fingers motion on the keys.

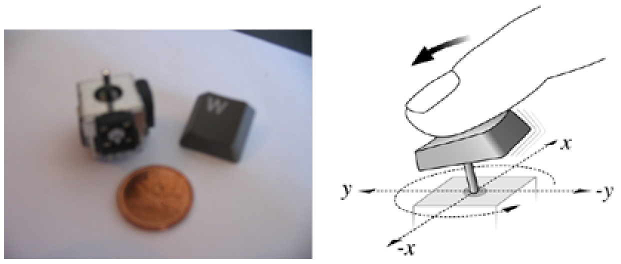


Figure 4. The mini-joystick sensor.

JOYSTICK KEYBOARD

Mini-Joystick Sensor

The mini-joystick sensor from CTSCorp [3] is used in our system. The sensor measures 2.07 x 2.05 x 1.33 cm (Figure 4). The sensor stick height is 0.6 cm and can be tilted a maximum distance of 0.5 cm from the center in any direction. The stick allows for 360 degree of rotation around its axis in clockwise and counter clockwise directions. The sensor has a momentarily switch, placed in the bottom of the compartment, and is activated by pressing the stick downward 0.1 cm.

Keyboard

The prototype has 54 mini-joystick sensors (Figure 5) laid out in 5 rows. Each sensor has a keycap installed on top. The key caps used are standard laptop caps with a groove that fits tightly on the sensor's stick. Enough space was left between keys to allow for any two adjacent keys to be tilted horizontally or vertically towards each other without blocking. The distance between the geometric centers of any two keys is 2.22 cm. This distance is 20% larger than the typical distance between keys in conventional keyboards (1.8 cm). The keytops were organized according to the QWERTY layout.

Circuit Board

The board has 2 sets of MUXs: analogue and digital (Figure 6). The analogue MUXs read the X,Y value of the stick position while the digital MUXs read the switch. A 3-bit address drives the MUXs to fetch the data from the sensors, requiring 8 cycles to read all the keys.

Controller

Conventional mice have frequencies that range from 30Hz for a low-end PS/2 mouse to 128Hz for a USB mouse. In order to support using any key in the Joystick Keyboard as a mouse, we needed a frequency of at least 30Hz for each key. This required a high sampling rate for the whole keyboard. We encountered preliminary difficulties achieving such high frequency using microcontroller chips. That is why; we decided to use the labjack UE9 [7], a high-end controller, in sampling the keyboard sensors. The labjack controller has a dual-processor design with 168

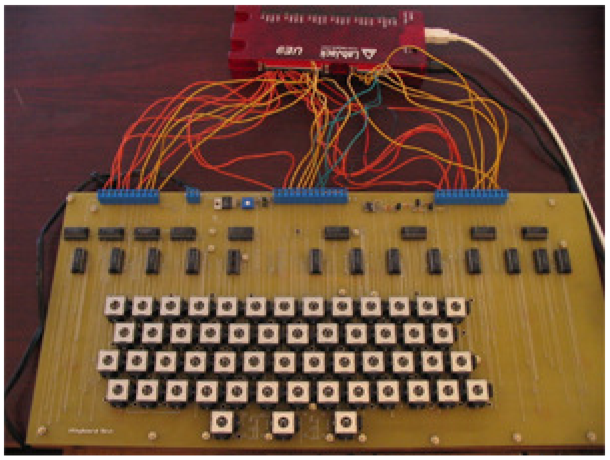


Figure 5. Custom fabricated circuit board.

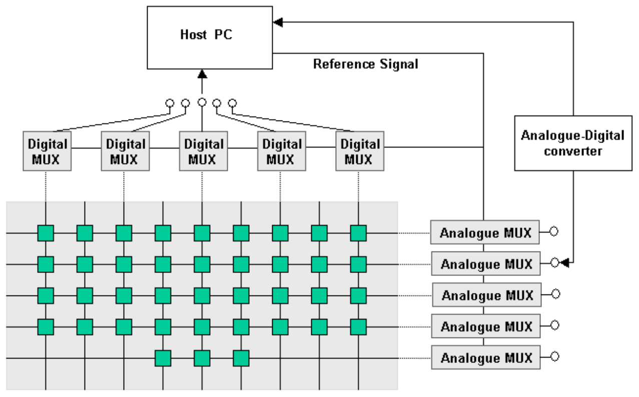


Figure 6. The Joystick Keyboard circuit configuration.

MHz of total processing power and supports both USB and Ethernet connectivity. One microprocessor generates square-wave signals and controls the I/O port, and the other microprocessor with a built-in A/D converter measures the values of the received signals and transmits them to the host computer. Connecting the labjack to the host using an Ethernet connection and using 12-bit resolution samples, we obtained a frequency of 40 Hz. Hence, each of the 54 keys is scanned 40 times a second. We believe we can achieve higher frequencies by optimizing the circuit design. However, 40Hz per key resulted in an acceptable performance for the purpose of our prototype.

SOFTWARE ARCHITECTURE

A software driver has been implemented that acts as a broker between the labjack/sensors and system applications (Figure 7). The driver continuously receives sensor data packets from the labjack and converts the raw voltage data into logical form. System applications can access the keyboard status by subscribing to the driver's events queue and receiving messages. Similar to a conventional keyboard

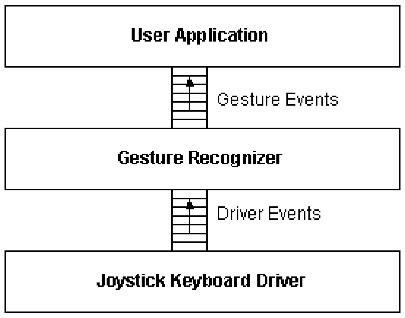


Figure 7. The Joystick Keyboard software architecture.

driver, the Joystick Keyboard driver reports changes in the status of pressed and tilted keys. A gesture recognition library has been implemented that process events from the keyboard driver and recognizes the gestures being performed by the user. User applications subscribe to events from the Recognizer library.

BASIC INTERACTION ISSUES AND FUNCTIONALITY

Tilt Inactive Zone: we have defined a tilting inactive zone, which covers 25% of the total distance the stick travels (Figure 8). Tiling the stick in this zone does not trigger events and hence does not move the cursor pointer. This shields the user from accidentally triggering a cursor move event during typing.

Pointing: for the purpose of this prototype, we implemented a simple linear mapping function. Tilting a single key will move the cursor in the same direction; north will move the cursor up; south will move the cursor down, east will move the cursor right, west will move the cursor left and so on.

Selection: to execute a “mouse click”, the user presses and releases any key in rapid succession.

Double-Click: to execute a “mouse double-click”, the user presses and releases any key in rapid succession twice.

Drag-Release: to execute a “mouse drag”, the user presses down a key while tilting another to move the selected object in the target direction. To execute a “mouse release”, the user releases the pressed key.

GESTURES

One Finger Techniques

Rotate: To rotate an object, the user rotates a key in the desired direction; rotating the key clockwise will rotate the object clockwise. Rotating the key counter clockwise will rotate the object counter clockwise (Figure 9a). The change in rotation angle is calculated by the change in angles from every two successive events.

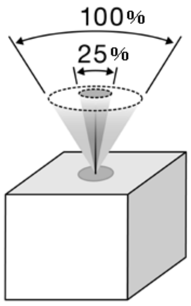


Figure 8. Tilt Inactive Zone.

Two Finger Techniques

Scale Up/Down: scaling up is achieved by tilting any 2 keys away from each other in the vertical, north/south, direction (Figure 9b). Scaling down requires an opposite gesture, where 2 keys are tilted towards each other (Figure 9c). We have chosen those two sequences because they resemble the actions of compressing and stretching an elastic object.

Zoom In/Out: zooming in is achieved by tilting any 2 keys away from each other in the horizontal, east/west, direction (Figure 9d). Zooming Out requires an opposite gesture, where 2 keys are tilted towards each other (Figure 9e). We have chosen those two sequences because they relate to the human field of view and thus are easier to remember by the users.

Scroll Left/Right: scrolling left is achieved by tilting any 2 horizontally adjacent keys towards the left (Figure 9f). Scrolling right requires an opposite gesture, where 2 keys are tilted towards the right (Figure 9g).

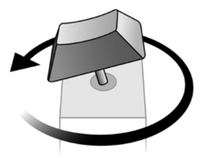
Scroll Up/Down: scrolling up is achieved by tilting any 2 horizontally adjacent keys towards the north (Figure 9h). Scrolling down requires an opposite gesture, where 2 keys are tilted towards the south (Figure 9i).

Three Finger Techniques

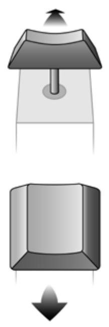
Rate-based Operations: in our two finger techniques, the rate of the operation being executed is constant. To directly control the rate of the operation being performed, the user can tilt any other key in the north/south direction. This gesture is supported for scaling, zooming (Figure 9j) and scrolling. We have also implemented this gesture for rotation (Figure 9k).

Four Finger Techniques:

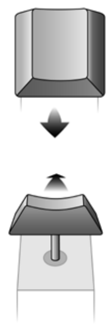
Polygon Selection: we have implemented a selection technique, which enables the user to define a polygon selection area. The polygon is defined by vectors, pointing outward from the current cursor location (Figure 10). Each vector magnitude and direction is controlled by titling a key. In this prototype, we have restricted our implementation to quadrilaterals



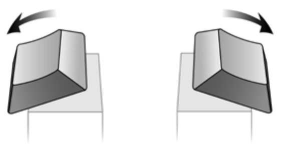
(a) Rotate Gesture.



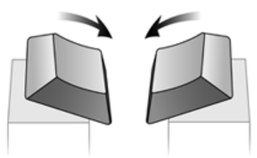
(b) Scale Up Gesture.



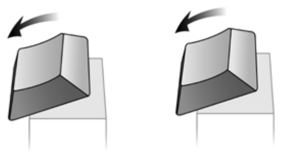
(c) Scale Down Gesture.



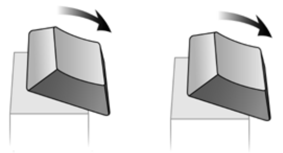
(d) Zoom In Gesture.



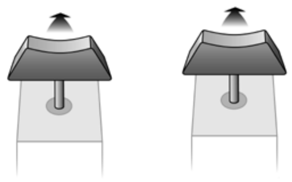
(e) Zoom Out Gesture.



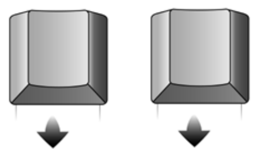
(f) Scroll Left Gesture.



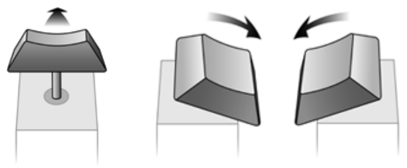
(g) Scroll Right Gesture.



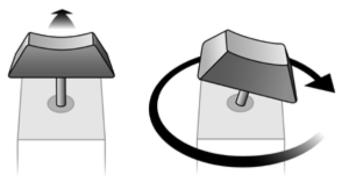
(h) Scroll Up Gesture.



(i) Scroll Down Gesture.



(j) Rate-based Zoom Out Gesture.



(k) Rate-based Rotation Gesture.

Figure 9. Joystick Keyboard Gestures.

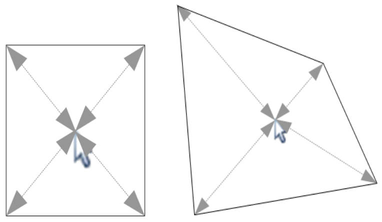


Figure 10. Sample selection areas created by tilting four keys.

Complex Gestures: our system supports any combination of the following gestures: zooming, scaling, and scrolling. This enables the user to perform two operations at the same time by tilting four keys simultaneously.

There are probably many other actions or functions representable by multi-finger gestures, for example, those based on the physical distance between tilted keys and those based on mappings between key locations on keyboard and object locations on screen. However, we have restricted our gesture set to basic object manipulation.

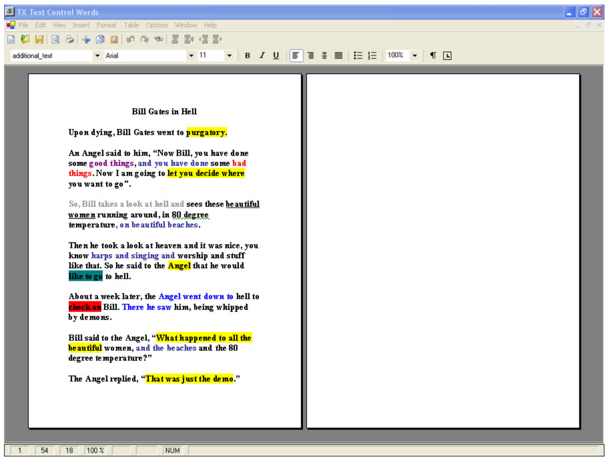
INFORMAL USER FEEDBACK

We conducted informal usability studies of the system using two applications: a text editor and a jigsaw puzzle. Ten users participated in our studies. Six of them were touch typists and all were laptop users. There were two sessions, one hour each, held in two consecutive weeks. At the beginning of each session, participants were informed about the nature of the research. After being introduced to the application, they were then shown each of the interaction techniques while also receiving details of how the gestures function in the context of the application. While our techniques are prototyped within these particular applications, there is no reason why they cannot be more generally applied to any other.

Text Editor:

To perform a basic evaluation of our design, we implemented a simple text editor application (Figure 11) using a plug-in editor software component [16]. The editor had two panels; read-only text was rendered in the left panel while the right panel was initially empty. The text, in the left panel, was a single page and written using several fonts, colors, sizes and highlights. The user task was to reproduce the left panel text in the empty right panel and apply the same visual settings. We have disabled some of the toolbars to force the user to navigate the menus and hence use more pointing operations. Each user was given 5 different single-page texts to reproduce.

We observed that the participants required little practice to learn the basic operation of the Joystick Keyboard, and were able to use it to accomplish the given task. User feedback was quite consistent among the ten participants. Following are the key observations:



Bill Gates in Hell

Upon dying, Bill Gates went to purgatory.

An Angel said to him, "Now Bill, you have done some good things, and you have done some bad things. Now I am going to let you decide where you want to go".

So, Bill takes a look at hell and sees these beautiful women running around, in 80 degree temperature, on beautiful beaches.

Then he took a look at heaven and it was nice, you know harps and singing and worship and stuff like that. So he said to the Angel that he would like to go to hell.

About a week later, the Angel went down to hell to check on Bill. There he saw him, being whipped by demons.

Bill said to the Angel, "What happened to all the beautiful women, and the beaches and the 80 degree temperature?"

The Angel replied, "That was just the demo."

Figure 11. Text Editor. Using the Joystick Keyboard, users were able to type and point.

Typing: users complained that typing requires more force than traditional keyboards. The mini-joysticks were designed to support clicking and not typing. This is a technology limitation that can be easily resolved by using an internal spring that requires less operational force.

Drag-Release Operation: while tilting a key, sometimes the user would press downward causing the switch to close. Although the application ignores key presses from a key that is already tilted, yet the sound of the click was annoying to the users, as it implied that a letter was typed.

Joystick Operation: the stick sometimes does not return to its initial position because of internal mechanical friction. This caused an error in our calculation of the tilt inactive zone. We have corrected that manually by inserting a pause in the study and recalibrating the sensors.

Tilt Inactive Zone: 2 users frequently tilted the keys beyond the inactive zone, triggering an un-intended cursor move event during typing. This suggests that the inactive zone should be dynamically adapted to individual users instead of having a global setting.

Avoided Keys: we noticed that users rarely tilted the spacebar, enter and backspace buttons even if it was the last key pressed and a finger is still positioned over it. One user commented that the "*space bar is too big to tilt*". Another user mentioned that he avoided tilting the backspace because he thought he "*might accidentally delete characters*" if the switch is pressed.

Switching Time: most users were hesitant when they started using the Joystick Keyboard, and switching between typing and pointing was done with great caution. However, during the short period of the study, we noticed that for most users the switching time seemed to decrease, and this was more noticeable with touch typists.

Sensor-to-Sensor Distance: none of the users felt that the extra distance of 0.42 cm between the keys is obstructive to their work.

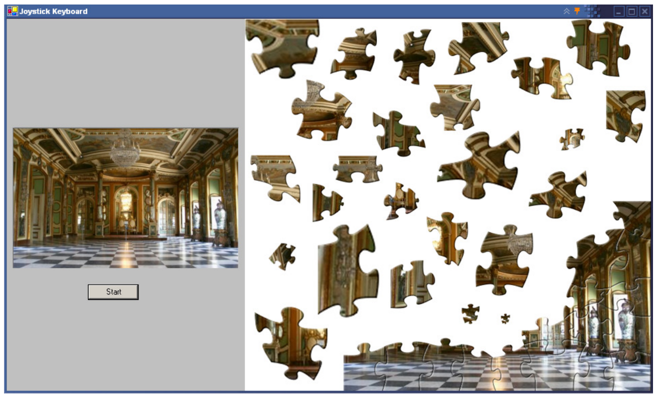


Figure 12. Jigsaw Puzzle. Using the Joystick Keyboard gestures, users were able to zoom in/out, scale up/down, scroll, drag and position pieces.

Jigsaw Puzzle:

In the second stage of our informal usability tests, we implemented a Jigsaw puzzle (Figure 12) to evaluate our gesture set. The puzzle scatters an image into 36 interlocking pieces and the user task is to reconstruct the image. During initialization, the application randomly scales and rotates the pieces to invite the user to use the scaling, zooming, and rotation gestures. Once the user connects two matching pieces, they are locked together. In this prototype, we have restricted our implementation of polygon selection to quadrilaterals (Figure 13).

We observed that the participants required little practice to learn the gesture set and were able to use the gestures effectively to solve the puzzle. User feedback was quite consistent among the ten participants. Following are the key observations:

Complex Gestures: 3 users tried to use 4 keys simultaneously to control an operation and it's reverse (e.g. scaling up and down). We did not anticipate this and the results were confusing to the users. We believe that when acting on a single object, only mutually exclusive complex gestures should be considered valid.

Polygon Selection: although most users liked the polygon selection feature, yet 3 users felt that it is not very helpful because they had to position the cursor inside the area of selection before invoking the gesture. 1 user described it as "*difficult to perform*". Also, we noticed that 3 of the 10 users always used 4 adjacent keys to perform this gesture. This suggests that they were trying to establish a frame of reference to guide them in the operation.

Tilting: 2 users commented that "*tilting a key forward is easier than tilting it backward*". The keycaps we used were slightly slanted towards the front, which might have contributed to this effect. However, we believe that this is an issue that requires a more in-depth investigation.

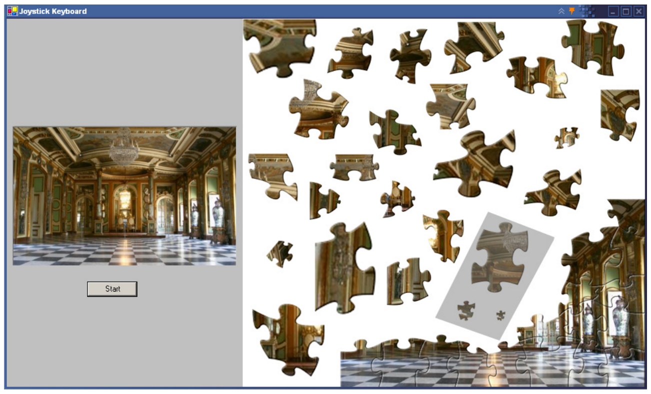


Figure 13. Using polygon selection, users were able to make more sophisticated selections.

FUTURE WORK

Although our early prototype shows potential, the formal usability tests are yet to be conducted. We are very interested in measuring the quantitative performance of tilting several keys at the same time, and ways to optimize the work of muscle groups involved in typing and pointing with such a design.

A great deal more work needs to be done on interaction techniques to map the device DOF's to other tasks of interest. For example, we feel that the Joystick Keyboard may enable interesting new 3D navigation techniques. A multi-DOF device like the Joystick Keyboard might enable new sets of 2D and 3D manipulators.

We are also interested in designing other keytop layouts, especially for small and handheld devices. For example, even with a very limited number of buttons, it would provide a rich selection of gesture commands. For example, several letters could be assigned to the same key and using tilt information to chose from them.

Using non-alphabetic languages, e.g. Chinese, with conventional keyboards is challenging because of the shear number of logographic characters in the language body. Although, input systems have been developed, e.g. pinyin, that simplifies the input mechanism, yet, users still have to do the input in two steps: the user type in some representation of the character, and then the system present a selection of matching characters for the user to select from. One possible solution is to display the pinyin choices in a pie menu and the user tilt the key to select the desired candidate. We plan to investigate how the Joystick Keyboard could be utilized to speed up the input of such languages.

CONCLUSION

This paper presented the Joystick Keyboard, a new multi-finger input device that combines typing and pointing. We have also presented several interaction techniques using two applications: a text editor and a jigsaw puzzle. Our initial studies show that this design is promising as a new integrated pointing and typing device.

REFERENCES

1. Card, S. K., Moran Thomas P., & Newell, A. (July 1980). The Keystroke-Level Model for User Performance Time with Interactive Systems. Communications of the ACM, 23 (7), 396410.
2. Card, S. K., Moran, T. P., & Newell, A. (1983). The Psychology of Human-Computer Interaction. Hillsdale, New Jersey: Lawerence Erlbaum Associates.
3. CTS Corp, mini-joystick 252 sensor, http://www.ctscorp.com/
4. Dietz, P., & Leigh, D. (2001). DiamondTouch: a Multiuser touch technology. ACM UIST Symposium on User Interface Software and Technology. 219-226.
5. EOWave, mini-joystick EOS10 sensor, http://www.eowave.com/
6. FingerWorks Inc, FingerBoard http://www.fingerworks.com/
7. Labjack Systems, UE9 controller, http://www.labjack.com/
8. Malik, S., Laszlo, J. (2004). Visual Touchpad: A Two-handed Gestural Input Device". In *Proc. ICMI-2004*, ACM Press, 289-296.
9. Mysliwiec, T. (1994). FingerMouse: A Freehand Computer Pointing Interface. *Technical Report VISLab-94-001*, University of Illinois Chicago.
10. Rekimoto, J. (2002). SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. ACM CHI Conference. p. 113-120.
11. Rekimoto, J., Ishizawa, T., Schwesig, C., & Oba, H. (2003). PreSense: Interaction Techniques for Finger Sensing Input Devices. In *Proc. UIST'03*, ACM Press, 203-212.
12. Rekimoto, J., (2003). ThumbSense: Automatic input mode sensing for touchpad-based interactions. *In CHI 2003 Late breaking*.
13. Rutledge, J, Selker, T. (1990) Force to motion functions for pointing, In *INTERACT'90: Proceedings of Human Computer Interaction*, 701-705.
14. Smart Technologies, DviT, http://www.smarttech.com/
15. Tactex Inc, Smart Fabric Technology, http://www.tactex.com/
16. Text Control Inc., TX Text Control .NET, http://www.textcontrol.com