PinkyMenu: Gesture with Pinky Finger Based Sketching Application for Tablet Computers

Cheng Zhang
Ubiquitous Computing Group
Georgia Institute of Technology
chengzhang@gatech.edu

Brandon Conway

Georgia Institute of Technology
bconway3@gatech.edu

ABSTRACT
In this paper, we present PinkyMenu, a new drawing application designed to explore integrating pen and touch interaction with the dominant hand to provide a more natural and enjoyable experience for novice and expert users alike. PinkyMenu allows for the pinky finger of the dominant hand, or any other finger, combined with a pen to control a two-level menu to manipulate the application. The major contribution of this tool is to explore the use of the dominant hand assisting with pen interaction to create a new way of developing menus and navigation systems without the use of a traditional menu and the non-dominant hand. In the end, we would like to show that this system could be expanded to several different kinds of systems.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors (Your general terms must be any of the following 16 designated terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification)

Keywords: Guides, instructions, formatting

INTRODUCTION
Pen interaction remains an important research issue for HCI researchers to examine. Even as new forms of interaction, such as finger-touch interaction and voice recognition technology, become more prevalent; pen interaction still maintains several advantages worth exploring. For starters, using a pen to write and interact with screens is much more natural and fits with the traditional model of writing with pen and paper. Pen interaction is also deemed to have higher accuracy as opposed to direct hand interaction with a tablet screen. However, we believe the most compelling reason is that nearly everyone is able to use pen regardless of age, profession, and education level. Meanwhile, it would not be as easy for most users to pick up two hand or multi-finger input in the same timeframe due to existing mental models and years of tradition.

Historically, improvements to pen interaction have developed in one of two ways. One is to make the pen more intelligent, meaning that there is additional functionality added to the usual tablet pen. An example of this is a pen analyzing a person's handwriting and assistance with forgeries [1,2], or attaching more hardware on the pen, such as a multi-touch pen[3,9]. Another way to integrate the pen and touch interaction is to attempt to simulate the normal interaction between pen and the non-dominant hand. From this point, developers would take this knowledge and try to apply it to a pen on a tablet computer. Even in regular writing, we need the non-dominant hand to assist in various tasks with dominant hand. For example, when people are drawing pictures on paper, the paper may be rotated by the friction generated between the brush and the paper. Usually, in such a case, most people tend to use their non-dominant hand to slightly rotate it back or press tightly on the paper, in case the paper shifts.

Despite these advantages, touch interaction is even more popular in the current technology industry and market. As small computers, such as smart phones and tablets become more prevalent, the use of touch screens continues to rise. One reason for this is that users need not purchase additional equipment because they are able to do any interaction with just their fingers. Brandl[4] makes a comparison on the strengths and weakness between pen-interaction and touch interaction. Generally, the author finds that pen input is good for high precisely input, does not have as many issues with occlusion, and is easier for user to leverage their mental model with pen and paper. However, touch input is quite complementary to pen-interaction. Because it supports the multi-points input, low attention input and no need for additional device. But, it lacks input accuracy and it easily suffers from occlusion problems.

Given all of this information, we have several areas we would like to approach in our project. At the base level, we seek to create a new drawing and sketching application for Android devices, and eventually other tablets. Beyond what you would normally see in an application like this, we

have drastically reduced the need for menus. This was accomplished by enabling the pinky finger on the dominant hand to assist the pen, along with several gestures to replace the functionality in many of the menu systems in sketching applications. Any menu we use do not take up screen space right away. Instead, they are called via a drop-down menu that only appears when a second interaction point is placed upon the screen. This allows for easier traversal of menus, and quicker response times. There are several benefits of this menu. Firstly, the user does not need to move their eye focus and the hand to access the menu, since the menu will be dropout around the stylus. Secondly, all the functions could be done by dominant hand. Besides, we do not drastically break the work flow, which can be important for someone creating artwork. We hope to show that this work can in fact be expanded to new areas in the field and improve our own drawing application.

RELATED WORK

Several previous works have provided us inspiration for our project. Hinckley's paper on creating new tools through pen and touch interaction gave us our initial burst of inspiration. We believe that much like the *Deskterity*, we can create new tools that could benefit artists and casual users alike [13,15]. We do note that we focus more on the sketching and drawing, while *Deskterity* is more of a general purpose "notebook" application.

Yee's paper on Two-handed interaction also gave us several ideas. However, the paper is from 2004 and many of the concepts are now out of date. What interested us came from taking the ideas of the paper, such as creating a pen and touch sketching/drawing application, and evolving his ideas to go along with the current research in the field[10]. In a sense we want to combine Yee's ideas with the ideas presented by Leitner in his paper "FLUX- A Tilting Multi-Touch and Pen Based Surface". In this paper, Leitner wants to create a surface capable of supporting both multi-touch and multi-pen based applications[12]. Essentially, combining the interaction focus of Yee and the abilities of Leitner's surface, we can create a multi-touch based sketching application.

Many ideas we hope to implement regarding the hand gestures (both dominant and non-dominant hands) were inspired by Wu's paper on gesture recognition. In this paper, Wu develops a set of design principles that can be used and steps to creating intuitive and successful gestures[11]. Wu's paper, however, only focuses on the interaction with the non-dominant hand, while we hope to extend this to pen and touch, and dominant hand usage.

Brandl's paper entitled "An Adaptable Rear-Projection Screen Using Digital Pens and Hand Gestures" fills in a lot of the holes regarding current research on interactive displays. It has given us good ideas we may want to implement regarding tracking in our system since the system reported great results with accuracy [14].

Another paper presented by Mathias Frisch, entitled "Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces" provided additional feedback on how best to study the gestures and tasks we would like to implement. While Frisch's paper focuses more on the editing of objects instead of creating, he runs several studies to learn what type of gestures and which method of creating gestures is best used[7]. He tests several types of gestures using one hand interaction, pen and touch interaction, and two hand interaction. We hope to implement all three of these in some way, so this paper serves as a great reference to how we can use this in object creation.

A lot of previous research has been performed in the area of pen interaction and the combination of pen and hand interaction. Whereas previous research usually focuses on two hand interaction to define separate functions for each hand, as far as we know, no research has explored the combination of pen input in conjunction with input from the dominant hand, especially the pinky finger. This avenue of research, we believe, provides a new direction for the usual paint application.

DESIGN

Overview

In this project, we implement a drawing application with several common functions on the Android Tablet, in this case the Nexus 7, which supports multi-touch. In addition, we built a two-level menu that can be manipulated by adding a second interaction point, for our purposes the pinky finger, along with several multi-touch based gestures. This allows the user to use the pinky finger to assist with the drawing application.

Our entire application is built from scratching, from the underlying paint application to the gesture recognition system. While several sources served as inspiration, we built our application from scratch to give us more flexibility when designing the application. The Android Software Development Kit includes standard multi-touch gesture recognition ability, but does not provide the flexibility to fit our need.

Two-Level Menu Design

We have seen many ways in which we can improve group productivity, through Smartboards and Tivoli style applications [17]. However, as systems get smarter, we need to be able to find a way to quickly express ideas or do complex manipulations without traversing through all of the menus in many modern day systems. The PinkyMenu system maintains a simple, yet elegant design that should be easy to learn for almost all users. The application itself has very little to the interface; in fact, the user is initially presented with a blank screen. When the user first draws, they just have a basic white brush stroke (See Figure 1).All menus can be activated and selected by placing the pinky finger on the screen.

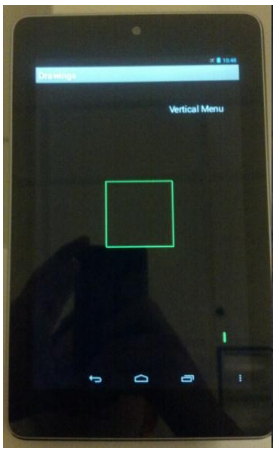

Figure 1 Application Screen with added Rectangle

We implemented a drop-down menu system, primarily because we were concerned with the stoppage of work when it comes to drawing and sketching. When an artist paints, the only time they need to stop is to change colors or brushes. We wanted to simulate that feeling as much as possible. When the user places the second interaction point on the touchscreen, our two-level menu system will pop up under that finger. A common problem in touch interaction is the fat-finger problem. To account for this, the positions of our menu are well-designed for a right-hand user in our current system. The first level menu is a vertical menu (Fig. 2) that pops up to the left of the stylus. The second level menu will appear above the stylus once a first level option has been selected.

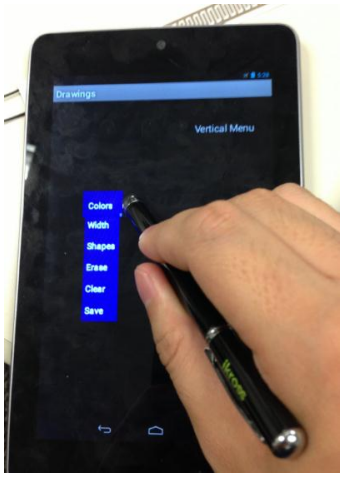

Figure 2 First Level Menu

The first level menu system currently consists of six options "Colors, Width, Shape, Erase, Clear, and Save". If a user selects an option, such as "Color", the second level will open up and give the color options. By drastically reducing the time to change options in menus, we believe we provide a system that will prove to be more efficient to use in the long run. While we could not completely take out a menu system, we believe the multi-touch drop-down system will reduce the amount of time spent stopping work to change items dramatically.

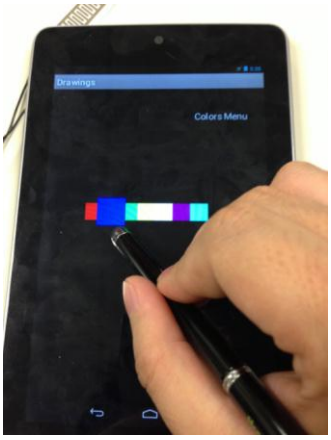

Figure 3 Second Level Menu for Color

The particular functionality of such features as "Undo" "Clear" and "Redo" work by implementing a pair of stacks. Any action a user performs is placed on an initial stack and the "Undo" ability becomes active. Once a user opts to undo an item, the item is then taken off the top of the initial stack and placed on what we call the redo stack. Clear works in a similar fashion, but instead of taking one item off the stack all the items are taken off, in order, off of the stack. Looking at the "Redo" button, as opposed to "Undo" and "Clear", this button is only activated once a user has removed any item from the screen. As soon as something goes on top of the redo stack a user can properly use the redo gesture. An item that is redrawn on the screen is taken off the redo stack and placed back at the top of the initial stack as the most recently completed action.

Gesture Design

There are two sets of gestures in PinkyMenu. The first set of gestures is designed to control the two-level menus. The second set completes the common functions, such as undo, redo, etc.

Gestures for Two-Level Menus

a. First Level Menu Activation Gesture

In order to activate the first level menu, the user needs to keep the pen stationary and move the pinky finger vertically. The degree can vary between 0-15 degrees. Moving the finger up and down will change the selected menu item (Fig. 4).

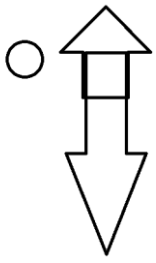

Figure 4 Vertical Menu Gesture

b. Second Level Menu Activation Gesture

To activate the second level of our menu system, the user can lift their finger to select the first level menu, and then place the finger back down moving it horizontally or in an arc to select an item from the second level menu(Fig.5).

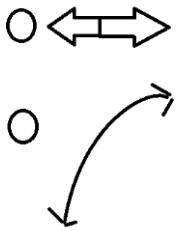

Figure 5 Second Level Menu Gesture

c. Gesture to activate the First Level Menu and Second Level Menu continuously.

Some more experienced users may not want to go through the routine of stopping and releasing their finger to move between menus. In this case, the user can just move their pinky finger vertically to select the first level menu and move straight to the second level menu by changing the direction in a diagonal fashion(Fig.6). If the selected first level item has a second level menu that menu will be activated immediately.

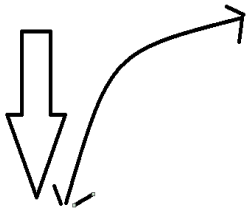

Figure 6 Two-in-One Gesture

Gestures for Paint Application Function

To further fix the issue of work stoppage, we added two sets of natural gestures to allow for quick manipulation of the system. Our gestures were inspired by the ideas behind the MacBook Trackpad. When using a browser, such as Google Chrome or Safari, a user can swipe two fingers to the right to move back a page or two fingers to the left to move forward a page. We map those same gestures to "Undo", moving two fingers to the right(Fig.8), and "Redo", moving two fingers to the left(Fig.7).

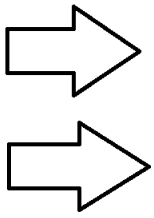

Figure 7 The Gesture to Redo an action

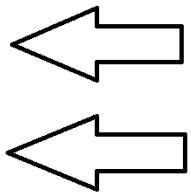

Figure 8 The Gesture to Undo an action

What this allows is for easy correction of mistakes a user may make, or if they decide they like the change to place it back in the work. Our second set of gestures again comes from the gestures used in internet browsers. When a user wants to scroll down a page on the Mac, all they have to do is pull two fingers up on the trackpad.

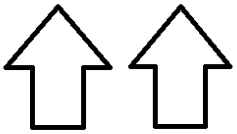

Figure 9 Gesture to move to next canvas

If they want to scroll back up, then just pull two fingers back down on the trackpad(Fig.9 and Fig.10). We wanted to implement something similar here for our application. Users should be able to flip back and forth between different canvases, much like someone who sketches can. Thus, when a user places the pen and a finger on the tablet, they can pull two fingers down to open up a new canvas. If they are ready to move back to previous work, it just requires pulling two fingers down instead.

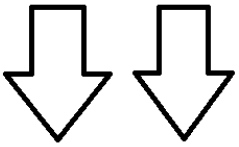

Figure 10 Gesture to move to a previous canvas

In the proposal, we proposed that certain actions, such as "Erase" and "Clear" could be implemented in the set of gestures we create. In the end, we opted against this for several reasons. For starters, this being a drawing application, we did not want to interfere with anything the user may be doing at any time in the process. There should be minimal chance of accidentally clearing the screen, when a user is just trying to draw a picture. In addition, being that the menu is also summoned by multi-touch, we did not want to interfere with the users calling of the menu, by adding ambiguous gestures. For these reasons, we opted to keep certain functionality inside of the menu system.

Gesture Recognition

Due to the nature of gesture software, our built from scratch gestures are limited by the information we receive from the system. Since we use an Android Tablet to develop the application, the information we collected from the device included: the number of touch points, the position of each touch point, the time of touch interaction etc.

In our system, most of the gestures are recognized by the angle or the distance of movement among different touch points. In the following part, we will describe the recognition algorithms for each gesture.

Gesture to activate the Two-level Menu:

The gestures for two-level menu are mixed gestures between pen and pinky finger. To make sure these gestures will not be mixed with our functionality gestures, which require movement from both interaction points, the system will first also detect whether the movement of first touch point is limited into certain areas. The system will then calculate the movement of the second touch point to make sure it is larger than the certain threshold because we do not want a small shake movement by pinky finger to also activate the menu. In addition, the angle of the vertical movement will be calculated. If the angle is smaller than 20 degree, the gesture will activate the first-level menu which is vertical. Otherwise, the second-level horizontal menu will be drop out. After the activation of menus, the distance of the finger movement will be used to detect which menu item is selected.

Gesture to active First Level Menu and Second Level Menu continuously:

This gesture will require the first-level menu is already activated. Once the system detects that the horizontal movement and the angle between the initial input position and the current position is larger than a certain threshold, the second level menu will be activated immediately.

Undo and Redo, Canvas Move Gesture:

Similar to the menu gesture, the system will first examine whether the movement is larger than a certain threshold. Upon finishing that task, the system will check whether the difference between the angle of the pen movement and the angle of pinky finger movement is larger than 10 degrees. If the gesture is in fact larger, the system will check whether the angle is smaller than 45 degrees. If the angle goes beyond 45 degrees, then the system will detect the undo or redo gestures. Otherwise, the canvas move gesture will be activated. And the direction of the movement will determine which concrete commands will be executed.

Hardware

Our initial goal for this project was to build the project on iOS systems, such as the iPad. We ran into several issues in the very early steps of the development that prevented us from using this system. For starters, iOS development tools are limited to Mac users only. As neither of us had a Mac we could use for the construction of this application, it made it difficult to use this system. Additionally, the fees incurred to Apple developers were not something we wanted to deal with for a semester project.

Given that Android tools were simpler to get access to and use, we moved our project over to the Android platform, which turned out to be more successful. Our tablet of choice was the Nexus 7, but we are hoping to test soon on a 10" tablet to see how the interaction differs between the two.

In addition, we have to use a stylus or a modal pen to properly test our interaction. This allows us to properly interact with the system, and helped us figure out the gestures that depend on the pen being placed on the tablet, such as Undo and Redo.

If we can, we would like to look more in depth at gestures presented by Robert Zeleznik in his examination of multi-touch gestures in relation to math. While his paper focuses more on solving problems than anything else, he presents various methods for providing intuitive gestures that people of any age can learn [6].

Application Developed

As we stated earlier, our project was completed on the Android tablet. Doing such, we used Java as the programming language of choice, developed in the Eclipse Juno platform. In addition, we also need to add in the Android Software Development Kit (SDK) in order to properly utilize the Android tools. We could have built our application with HTML to allow for use on multiple platforms, but a native application has a better feel and works better with each individual platform. With the success of this project, we believe we can extend this application to other platforms in the future.

EVALUATION AND ISSUES

Upon completion of our system, we attempted to debug and improve it by testing on the Nexus 7 tablet computer. Using a 7" tablet presented several issues that we had not intended. The limited screen space presented issues when it came to placing a pinky finger on the screen. People with larger hands would have immense difficulty using the system. For this reason, we are hoping to soon test the application on a 10" tablet which would give us a wider space in which to test the feel of the application. An internal evaluation also has been done to improve the design such as find the proper threshold or parameters on gesture recognition and the design of the menu as well.

As for the functionality of the system, it proved to work very well. The first level menu responded to any and all kinds of commands. All brush strokes worked properly and increased the size of the next strokes. Erase mode properly turned the paint brush the color of the background, which allowed for easy erasing.

As for the usability of the system, we also found that the flexibility of people's pinky finger is different. Different people have different comfortable ways to move and con-

trol their pinky finger. And the accuracy of the movement by the pinky finger varies as well.

Other than the space issue, we found only minor issues with our system. The major issues came when testing the up and down gestures. The bitmap manipulation proved to be more difficult than intended, and the canvas did not always shift properly. We hope that this is an issue we can eliminate in the future. The only remaining issue was that when performing particular gestures, the pen will still draw a line as well. We had to make several workarounds in order to make our gestures work properly.

DISCUSSION

When developing PinkyMenu, we knew we would run into several obstacles since this is a fairly new avenue for implementing a paint program. The paint programs we investigated in the past stuck to the traditional menu styles seen in applications like Microsoft Paint. The big innovation seen in these applications is that the menus are placed on the screen for users to touch, as opposed to in the usual Android menu area. With no background research, we were essentially trying something new which came with a bunch of issues, expected and unexpected. We were uncertain how well the 7" tablet would work with the two touch input. While the hand can easily fit, draw, and access the menus in our system there were a lot of issues with occlusion that made us believe that a 10" tablet is truly the appropriate size for the application.

Another interesting debate revolved around what type of menu system should be used in the application. Initially the goal was to use a circle menu for our system. Circle menus have several advantages; many involve minimizing time in menus according to Fitts' Law. With our goal of minimizing time in menus, we believed that this approach might be correct. The more we worked with the system, the more we realized a traditional vertical menu would be an appropriate choice. With pen and touch interaction, using a circle menu inevitably means that some part of the menu would be occluded by the drawing hand. While a user could probably guess what is in place, we determined that our system had below average affordances as it was, we did not need to make the system any more complex.

While we tested the system internally, we found that people with low dexterity in their dominant hand occasionally had issues bringing up the menus. While they got better quickly, it was interesting to think about ways to improve our application to make it easier to bring up menus for those that do not have great dexterity.

Overall feedback on our work was good. While there is always room for improvement, there was great interest in hearing about the direction of our work and how this style of application could work in the future to improve the user experience with tablets. Users often mentioned how they did not care for the menu system for tablets, and this system would go lengths to improve the experience for them.

FUTURE WORK

As far as the actual PinkyMenu project goes, we would like to perform a well-designed user study to collect information on how people best move their pinky finger along with the precision and accuracy in which they can perform the tasks in our application. By answering these questions, the users will help us design more ubiquitous gestures with the pinky finger.

A great task would be to try and recognize pen input versus other parts of the hand. By recognizing different kinds of inputs, the gesture vocabulary could be dramatically increased. This would allow for us to create more scenarios and further eliminate the menu system.

We believe there is a great future in designing apps this way. Eliminating the current menu system can drastically reduce wasted time on computers, and allow for a quicker user experience. While we have only begun to scratch the surface of this style of interaction, it would not be hard for us to completely fit PinkyMenu with even more functionality, allowing us to implement more gestures to make our application more robust.

While there is more we can add, as we discussed earlier that we would like to move the application to different platforms. Porting the system to iOS systems, such as the iPad, and Windows 8 systems would allow us to test the functionality on different systems. Additional studies could then be performed, including the different uses of this platform between users of each platform.

After completing this application, it would be worth exploring other areas that could potentially benefit from this style of interaction. Photo editing software, such as Photoshop, could be made a lot more natural to use by eliminating big chunks of their menu system, and replacing them with gestures and multi-touch implementation.

SUMMARY

In this paper, we have addressed several obtainable goals for our application, plus a few that can be implemented if we have time, or in the future after the conclusion of class. We seek to create a new style of drawing and sketching application, one that could essentially end the days of needing a separate sketching tablet for professional users, and give novice users a chance to learn how to create higher quality work without the need for additional equipment. Our project seeks to simulate the flow of a normal artist by not hampering users with endless amounts of menus. We want users to feel like they are truly creating works of art with our system. It can be accomplished through the use of natural, intuitive gestures that do not disrupt the flow of work. This system enables the user to interact with the tablet with only one hand, further reducing the cognitive workload and physical movement of the hands. However, we do believe there will be instances where the user will have no choice but to use a menu. We hope to alleviate this issue by the use of drop down menus activated by the pinky finger. We believe this will be an essential way for the user to quickly make a choice and get right back to work. We have

defined several ideas we have for our application that take advantage of pen and touch interaction, plus use of new technology that could determine which part of our finger is currently touching the screen. The combination of these elements should create a wide variety of gestures that users could easily pick up and use to create sketches.

ACKNOWLEDGMENTS

We would like to thank Georgia Tech's Research Network Operations Center for providing us with the Nexus 7 and other hardware to test and debug our application.

REFERENCES

1. Jiang Y., Tian, F., Wang H., Zhang X., Wang X., Dai, G.: Intelligent understanding of handwritten geometry theorem proving. IUI 2010: 119-128.
2. Jiang, Y., Tian, F., Wang, X, Zhang, X., Dai, G., Wang, H.: Structuring and manipulating hand-drawn concept maps. IUI 2009: 457-462.
3. Song, H., Benko, H., Guimbretiere, F., Izadi S., Cao X., Hinckley, K. (2011). Grips and gestures on a multi-touch pen. Proceedings of CHI 2011, ACM Conference on Human Factors in Computing Systems. p. 1323-1332.
4. Brandl, P., Forlines, C., Wigdor, D., Haller, M., Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. AVI 2008, p. 154-61
5. Harrison, C., Schwarz, J. and Hudson S. E. 2011. TapSense: Enhancing Finger Interaction on Touch Surfaces. In Proceedings of the 24th Annual ACM Symposium on User interface Software and Technology (Santa Barbara, California, October 16 - 19, 2011). UIST '11. ACM, New York, NY. 627-636.
6. 6. Zeleznik, R., A. Bragdon, F. Adeputra, and H.-S. Ko. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In Proc. UIST'10. ACM Press. p. 17-26
7. Frisch, M., Heydekorn, J., Dachselt, R. Investigating Multi-Touch and Pen Gestures for Diagram Editing on Interactive Surfaces ITS '09 Conf. on Interactive Tabletops and Surfaces.
8. Tian, F., L. Xu, H. Wang, X. Zhang, Y. Liu, V. Setlur, and G. Dai. Tilt menu: using the 3D orientation information of pen devices to extend the selection capability of pen-based user interfaces. In Proc. CHI'08. ACM Press. p. 1371-1380
9. Song, H., Benko, H., Guimbretière, F., Izadi, S., and Cao, X. Grips and gestures on a multi-touch pen. Proc. of the SIGCHI conference on Human factors in computing systems, ACM (2011).
10. Yee, K.-P. Two-Handed Interaction on a Tablet Display. CHI 2004
11. Wu, M., Shen, C., Ryall, K., Forlines, C., and Balakrishnan, R. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. IEEE Int'l Workshop Horizontal Interactive Human-Computer Systems (TableTop). 2006.
12. Leitner, J., Powell, J., Brandl, P., et al. Flux: a tilting multitouch and pen based surface. Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, ACM (2009), 3211-3216.
13. Hinckley, K., et al. Manual Deskterity: An Exploration of Simultaneous Pen + Touch Direct Input. CHI 2010 Ext. Abstracts
14. Brandl, P.; Haller, M.; Hurnaus, M.; Lugmayr, V.; Oberngruber, J.; Oster, C.; Schafleitner, C. & Billinghurst, M. An Adaptable Rear-Projection Screen Using Digital Pens And Hand Gestures, In Proc ICAT 2007, IEEE Computer Society (2007), 49-54.
15. Hinckley, K., K. Yatani, M. Pahud, N. Coddington, J.Rodenhouse, A. Wilson, H. Benko, and B. Buxton. Pen + touch = new tools. In Proc. UIST'10. ACM Press. p. 27-36
16. T. Igarashi, S. Matsuoka and H. Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design", Proc. Of ACM SIGGRAPH, 409-416, Los Angeles, USA, Aug.1999
17. Moran, T.P., P. Chiu, and W. van Melle. Pen-Based Interaction Techniques For Organizing Material on an Electronic Whiteboard. In Proceedings of UIST '97. Banff, Alberta, Canada. pp. 45-54, October 14-17 1997
18. Vogel, D., Cudmore, M., Casiez, G., Balakrishnan, R., and Keliher, L. 2009. Hand occlusion with tablet-sized direct pen input. Proc. CHI'09. ACM Press, 557-566gel