Integrating Non-Dominant Hand and Stylus Interactions for Tablet Note-Taking Applications

Hsiao-Ching Su
Georgia Institute of Technology
Atlanta, GA USA
kelly.su@gatech.edu

Marites Hendrix
Georgia Institute of Technology
Atlanta, GA USA
mhendrix8@gatech.edu

ABSTRACT

Static toolbars for note-taking applications on tablets is commonplace, but takes up valuable screen space and forces users to conform to the application's needs rather than the application conforming to a user's needs. Current static toolbars with stylus options break user concentration and ignore the potential that the non-dominant hand can play in the tablet note-taking process. This project aims to take away the necessity of a static toolbar and gives the non-dominant hand a role to play in tablet note-taking. This project allows the non-dominant hand to perform touch commands that select stylus tool options, perform quick multi-touch taps for undo, redo and toggling the larger static toolbar, as well as work with the stylus to perform intuitive copy and paste gestures for selected drawings. This was accomplished on an Android tablet with a capacitive stylus, a Samsung Tab S2 tablet, and Android Studio.

Author Keywords
Tablet; Stylus; pen+touch; thumb; bimanual input; Touch; Sensing; Drawing; Note-taking

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

General Terms: Design, Human Factors

INTRODUCTION

A tablet is a dynamic tool that allows a user to perform tasks that laptops and phones are commonly used for, such as email checking and web surfing, but sets itself apart with the ability to write and draw comfortably onto the screen. As tablet user penetration worldwide has increased from 14% in 2014 to 18.9% in 2019, and over 32.2 million tablets shipped from just the first 6 months of 2019 alone (reference [10]), the importance of creating user-centered applications that focus on the tablet's capability of drawing and writing has become more relevant.

Note-taking tablet applications are particularly popular for students and those in the workforce. Current note-taking applications on tablets follow a similar standard layout: a static toolbar on one edge of the application and an empty canvas centered on the screen to write on. This layout has limitations that hinder productivity and take up valuable screen space.

With current note-taking application layouts, changing stylus input types forces a user to break focus from their current task to go to the toolbar and find the settings they need. Though note-taking applications such as GoodNotes on iPad will try to minimize this process by allowing users to create shortcuts to favorite settings options (i.e. color and stroke width), this physical movement to the toolbar still requires the user to break from what psychologists refer to as "flow" [2].

This project created a note-taking application that took "flow" into account with its design approach. To take flow into account, the note-taking application must 1) challenge and require skill, 2) avoid interruption, 3) allow users to maintain control 4) give the user speed and feedback, and lastly 5) transform time (a user in flow perceives time passing differently than out of flow) [2].

From previous research, it was found that a person that uses stylus and touch together for tablet-writing is ranked the highest by users in terms of their overall preference, ease of use, and accuracy compared to just pen or just touch inputs [3]. Combining touch and pen balances the pros and cons of each; for example, the single input point of the pen is balanced by the multi-touch abilities of the hand, the pen's high precision can be blended well with the natural feel of a less precise touch, and more [3]. This project combined the use of a stylus with the non-dominant hand to encourage this kind of interaction. Specifically, this project focused on integrating the use of the non-dominant hand in note-taking applications in a way that does not disrupt the user and frees valuable screen space when desired. Currently, the non-dominant hand has no part in the tablet writing stage, but the interaction between stylus and touch brings additional functionality to the writing experience in this project.

"Pen + Touch = New Tools", a research paper by Ken Hinckley, studied how users wrote with a pen and paper notebook [7]. This project accounts for the findings in this research. It was found that the non-dominant hand from users tends to arrange or hold objects and the pen tends to write. Design properties discussed of touch and pen includes contacts, occlusion, precision, elementary inputs of hands: thumb, palm, pinch, etc. Taking inspiration from previous work from Yang Zhang's Pen+Touch Interaction on Tablets [16], this tablet utilizes a finger from the non-dominant hand to bring up a quick radial menu for changing tools for the stylus, as well as different shortcut colors. Figure 2 from "Thumb + Pen Interaction on Tablets" [13] demonstrates three methods of thumb and pen interaction, which inspired part of this project's design. This paper described a thumb marking menu [9] with a spring-loaded mode [6] shown in Figure 6 of "Thumb + Pen Interaction on Tablets" [13], where the menu springs out after the non-dominant thumb holds down on the side edge of the tablet for a short time. The user moves their thumb along the menu to switch between different choices while the menu shows animation for the menu item being selected through a higher opacity or color change. Secondly, there is a holding spring-loaded mode [13], where the non-dominant thumb holding at the side edge calls up a menu at the stylus location, and the user uses the pen to make choice.

The design for this project was inspired by the works of Ken Hinckley, Procreate (an iOS art app) and GoodNotes. The application created for this project successfully implements new interaction techniques between the non-dominant hand and the stylus. The non-dominant hand can bring up a radial menu containing stylus tool options with any finger, the design of which was inspired by Ken Hinckley's work. Multi-touch taps of two fingers, three fingers and four fingers can trigger undo, redo and toggle the visibility of the static toolbar on the canvas respectively. This was inspired by Procreate and was seen as useful in the note-taking space. Lastly, a new copy and paste method that utilizes multi-touch was implemented into this project, driven by the desire for a more intuitive solution to an action commonly performed by students needing to redraw basic diagrams and schematics. This particular functionality aimed to satisfy the need for "flow" for users.

CURRENT NOTE-TAKING APPLICATION TECHNOLOGY

There are many note-taking applications available on iOS and Android platforms. On iOS, GoodNotes and Notability are the most dominant apps. Common features available are paper setting options, importing and exporting notes to/from different formats, synchronization to the cloud, searching written text, and sorting papers. Targeting Apple Pencil users, the tools in these applications include a shape beautification tool, pencil tool, marker tool, and eraser tool, all with color and size options [5]. Specifically looking at GoodNotes, the application inspiring this project, users can store their top three preferred colors into a frequently used palette. The lasso tool on this note-taking app is a selection tool where users can define the area of selection by free-drawing with the Apple Pencil. The elements within the selection area drawn can be moved, cut, copied, or adjusted in size and color after bringing up a separate menu on a finger hold.

Procreate is an iOS drawing application that has popularized the heavy use of gestures, particularly gestures involving multiple fingers [14]. Existing gestures in this app include a single finger tap and hold for a fully customizable quick menu, two-finger tap for undo, three-finger tap redo, four-finger tap toggle of the menu borders, and more. These gestures enable high user productivity and are starting to influence popular note-taking applications such as GoodNotes, who has just added two finger and three finger double tap undo and redo likely inspired by Procreate [4].

KeyNote, another note-taking application for iOS, has some unique object manipulation with touch, integrating features such as "constrain drag", allowing for object dragging in straight lines, or being able to select multiple objects through tapping with both hands if desired [1]. This partially influenced a new copy and paste method integrated into this project that involves multi-touch.

Gesture and pen/touch interaction is showing its way into the note-taking field, but is still dominated heavily by the drawing application market. The goal of this project is to start integrating the best and most used features in note-taking with the innovative gestures from artistic applications and existing note-taking applications.

NEW INTERACTION TECHNIQUES

The note-taking application developed for this project integrates the following: a circular tool menu that appears around the placement of a finger from the non-dominant hand on the screen, new copy and paste functions that use two fingers or one finger with the stylus for the select tool, and multi-touch integration for indicating undo, redo and toggling the visibility of the larger toolbar. This iteration of the application has been completed with Android studio with a Samsung Tab S2 paired with a capacitive stylus. This section will describe how each component was developed and how each function was implemented.

Application Hierarchy

The application code is divided into three classes: MainActivity, CanvasUI, and RadialMenu. The MainActivity class initiates the entire application, creating the main canvas object (CanvasUI), and the static toolbar. It lays out the static toolbar to sit at the top of the tablet and connects each button in the toolbar to their appropriate functions in the canvas object.

The CanvasUI object handles all touch events for drawing and handling multitouch, as well as the logic for manipulating Stroke objects drawn to the canvas. This canvas defines two other objects: Stroke and GroupCopy. A Stroke object is important to all drawing functionality in the application. Every time a user places down his/her stylus, drags, and lifts, the line created is a Stroke object. The Stroke object holds the Android native Paint and Path objects created by the user's drawing. It also contains the value of multiple boolean flags that are used for other functions created in the application, as well as flags that indicate what tool the Stroke is associated with. A GroupCopy object holds an array of Strokes and an Android native RectF object that defines the bounding box around all the strokes in that array. The GroupCopy object is only used for the copy and paste functionality.

The CanvasUI class extends the native Android View object. The logic for drawing and multi-touch quick functions is done by overriding the *onTouchEvent()* function associated with View objects. This event handler function is triggered by all touch related events for this project, including: drawing, bringing up the radial menu, triggering multi-touch functions, and performing copy and paste. By looking at the MotionEvent value created from the *onTouchEvent()*, the number of pointers on the screen and type of action performed can be identified by its value.

Lastly, the RadialMenu class draws the radial menu and handles the logic behind selecting color, stroke width, and tool type from this menu. This object is created within the CanvasUI and is queried by the canvas for the values selected by the user in the menu. The radial menu is passed coordinates of a touch from the canvas to handle where to draw the menu and to calculate stroke width value.

Distinguish Touch from Stylus

To accomplish basic drawing, the application needed to detect the difference between a stylus touch and a finger touch. Tablets with active digitizers, such as the Apple iPad or Microsoft Surface Pro, are capable of using active styli that, in software, are distinguishable from touch. The Samsung Tab S2 used for this application is a capacitive tablet and cannot use active styli, only passive styli. This means that both stylus and touch or anything that causes a change in capacitance on the screen is detected and looks the same in software [15].

Ideally, this application would have been created for a tablet with an active digitizer, but due to resource constraints, a capacitive one was used. To distinguish touch from a stylus, the touch radius value was used. In the *onTouchEvent()* function for Android View objects, the MotionEvent created from a touch contains a value for the radius of the touch. This value constantly changes, and for stylus and touch, the values can sometimes overlap. A running average had to be calculated to make the stylus vs touch distinction more accurate.

The longer a stylus or finger is held or dragged on the screen, the more touch events there are. A value called *touchAvg* was created and is reset on every new *ACTION_DOWN* event. An *ACTION_DOWN* event is the first event to be triggered on the placement of anything to the screen. As the user draws, the touch radius sampled is calculated as part of this *touchAvg* in the CanvasUI.

The CanvasUI object by default assumes that the touch is a stylus and will begin drawing immediately. If the *touchAvg* array has more than 10 samples and is still has an average value that correlates to the calculated stylus radius, then the canvas will keep letting the user draw.

If after 10 touch samples the touch radius average is instead identified as a finger touch, then the Stroke drawn is deleted and the CanvasUI switches to treating the touch as a finger. This tolerance of 10 touch samples is short enough to allow for a relatively accurate detection of touch vs stylus and also is short enough so that if a finger draws a line, it is barely noticed by the user before being removed. By default, the finger touch will trigger the radial menu to pop up, which is discussed in the next section.

With the distinction of stylus from touch made, basic drawing capabilities were able to be accomplished. The CanvasUI object holds an array that contains all Stroke objects that have been drawn, and this array is iterated through and drawn by overriding the *onDraw()* method of the class.

Thumb Tool Menu

A popular method of writing on a tablet is to have it propped on a stand so that the tablet is tilted towards the user and not entirely flat on the table. This means there is a natural feel to holding the tablet in the non-dominant hand while the dominant hand is writing. The tool described in this section is named the thumb tool menu due to the assumption that a user's hand would be resting on the tablet and the thumb would be the most accessible finger. However, this tool is not limited to the thumb, and in reality, there is no distinction between the thumb and any other finger. This means that if a user likes to write on a tablet entirely flat on the table, they can just hold any finger that feels natural to the screen to pull up the menu.

Upon the press and hold of any finger to the screen, a radial menu will appear centered on the finger's placement (Figure 1). Once the menu appears, as long as the finger is held down, this menu will not disappear and will hold its location until the finger leaves the screen. The options on this menu are the following: Pen, Eraser, Marker (for highlighter), and Select.

The Pen and Marker tools have three preset color options to choose from. To select a tool or color, the finger simply has to be within the angular bounds of the selection. This means that a user does not have to place their finger directly on top of any setting since a selection is calculated by the current touch's relative angle to the center of the menu. For example, if selecting Pen, drag the finger to anywhere within the angular bounds of the Pen section and further color selection can be made by moving the finger as well.

The RadialMenu class will begin drawing itself at the initial coordinates of the finger placement when the CanvasUI object tells it to. As the finger moves, the angle of the finger's current coordinates relative to the center is used to determine what "pie section" to darken when drawing to show what is currently selected. For Marker and Pen, if the finger is selecting one of those sections, a further menu is drawn with three circles showing color options. Only when the finger lets go of the screen do any of these selections get sent to the CanvasUI object.

In the original proposal, the static toolbar was going to have a larger selection of colors so that the user can choose which three colors for Marker and Pen would be shown on this menu, but due to time constraints, this was not implemented.

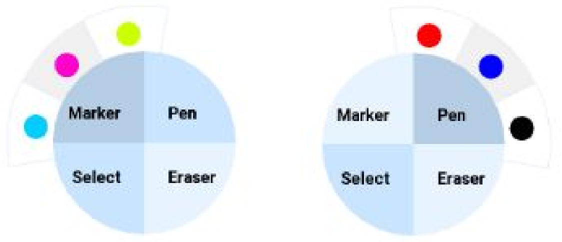

Figure 1: The left radial menu is being hovered on "Marker", and the right is on "Pen". The hovered tool is darkened and an arc-shaped color picker expands.

Additionally, for Pen, Eraser, and Marker, as a user's non-dominant finger is on the screen and displaying the radial menu, the tool's stroke width can be changed by dragging the stylus up and down on the screen (Figure 2). For example, if the user wants to have a red pen with a size of 10pt, then first the user will place their thumb on the screen. The user will drag their thumb to hover over "Pen" area, then drag over the red color section. As the user holds their thumb to the screen, they can take the stylus and drag vertically up or down anywhere on the canvas to change the tool's stroke width the same way a slider generally handles this. As the user slides their stylus up and down, the value of the tool width is shown numerically on the screen. Once the finger lets go of the canvas, the settings are sent to the CanvasUI object from the RadialMenu object. Selecting the Eraser is similar to Pen and Marker when changing its stroke width, but it does not have the additional arc-shaped menu for color. However, the Eraser does have a circle-shaped indication when the user draws an "Eraser stroke" on the screen (Figure 3), which Pen and Marker do not have.

This stroke width value is implemented by evaluating the touch event coordinate change in the vertical direction of the stylus. When the radial menu is showing, the CanvasUI object is switched to "menu mode". When in menu mode, if a second pointer appears, this triggers the RadialMenu object to begin calculating stroke width based on the second pointer. This second pointer is assumed to be the stylus as this is the natural option when writing. When the stylus is placed down, its y-coordinate position on the screen is defined as its initial position. As the y-value changes, the stroke width value increased or decreased by 5 pixels for every 50 pixels moved on the screen. The smallest value for stroke width is 5 pt and the largest stroke width allowed is 50 pt.

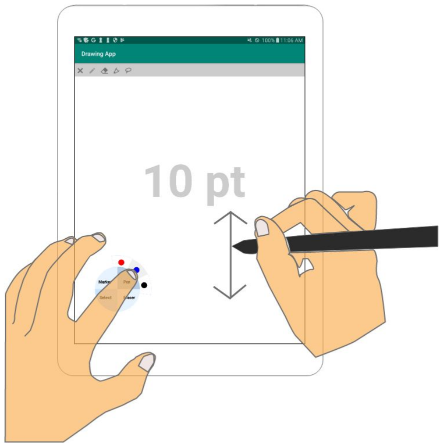

Figure 2: Pen and Marker tool with Radial Menu: Selecting the color by non-dominant hand and changing the stroke width by the stylus. The gray arrows indicate the stylus's movement on the screen.

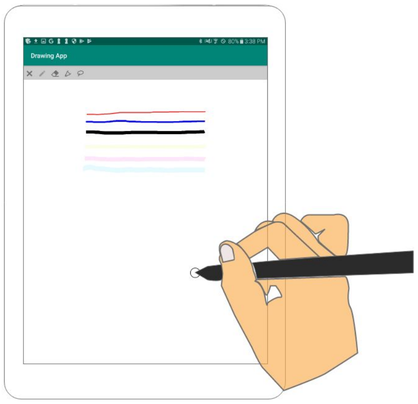

Figure 3: The circle-shaped indication of the Eraser tool, and the strokes of Pen and Marker in the middle.

Multi-touch Integration

The note-taking application looks out for multi-touch events from the non-dominant hand. While a single finger hold brings up the radial tool menu, two, three and four-finger taps are assigned different quick functions.

Inspired from Procreate, an advanced drawing application for the iOS, a two finger touch will trigger "undo". This allows a user to quickly undo something they just drew, saving time from having to select the eraser tool and dragging the stylus around manually for every small mistake. If done on accident, a three finger tap will trigger "redo". The undo/redo functionality is done by having the CanvasUI keep track of a “delete list” of Stroke objects. When undo is keyed, the most recent Stroke in the canvas stroke list is deleted and that Stroke is placed into the delete list. For redo, the most recently added Stroke to the delete list is added to the end of the Canvas Stroke list and removed from the delete list to be drawn again.

A four finger tap hides and shows the main static toolbar. Though the key to this application is the radial menu, a full toolbar should still be available so that quick colors and options can be set, and more advanced options can be selected. This project does not implement more options in the toolbar due to time constraints, but with more time it would. By allowing this toolbar to hide, this lets the user have the full screen be their canvas. A four finger tap is a fast way to bring the toolbar up, change some settings, and go back to primarily using the radial menu.

Every touch event in Android can be associated with a value. If the MotionEvent object action value is 773, it is associated with an *ACTION_DOWN* event with four pointers on the screen. This triggers the visibility toggle of the static toolbar. There are unique values for a two finger tap, three finger tap and a single pointer on the screen as well.

The functions associated with each tap are not directly called after the tap, however. While a four finger tap triggers the value 773, it also triggers the values for three fingers, two fingers and one finger at the same time. With a three finger tap, the values for a two finger and one finger tap is made as well. To not call multiple functions at once, the calling of a function is done after the *onTouchEvent()* is completed. The values created from the MotionEvent are processed after the full touch event so that only one function is triggered at a time.

Select Tool Expansion

Traditionally, the lasso tool in drawing applications lets a user free-draw around a group of objects to select them [8][11][12]. Different functions can be performed on the selected objects by right-clicking or tapping on the objects to open a menu. With note-taking, the lasso tool is popularly used to simply rearrange drawn objects or to copy and paste diagrams or drawings that need to be done repeatedly. Equations, diagrams, and drawings are commonly repeated when taking notes in class, so a faster way to complete this will be useful.

This project accomplishes a basic select tool that performs this same functionality, but also accomplishes a new way to copy and paste (Figure 5). Once the select tool is chosen from the radial menu, a user can drag a dashed square around the object they want to move or copy. Once all desired objects are at least partially within the rectangle drawn, a user can tap within that rectangle and the rectangle will snap to fit over all selected objects as seen in Figure 4. Then the user can drag the object on the screen to move it to another location with their stylus.

The first rectangle drawn is just the initial selection rectangle. To determine what objects are within the selection rectangle, each Stroke object’s Android Path value is approximated into an array of points. Each point is iterated over and checked to see if it is inside the drawn rectangle. If any point of a Stroke’s Path is within this selection rectangle, then the entire Stroke object is considered selected. After drawing the initial selection rectangle, tapping the stylus within that rectangle creates a new dashed rectangle that fits around all objects selected. This is done by calculating all the bounding boxes of each selected stroke and calculating a new bounding box that uses the max and min points from this array of bounding boxes.

The new copy function expands this basic selection functionality and works similarly to fanning cards out on a table. Once making a normal selection, copy and paste can be performed. The user can place the stylus onto the item being copied, and as they hold the stylus down, a finger can take and drag the object to the side. Copies of the drawing are stamped in increments and follow the path that the finger moves in. The longer and farther the item is dragged, the more copies are made. If the stylus drags out in a large "S" shape, then the copies will be stamped out in this "S" shape on the document, which is demonstrated in Figure 5.

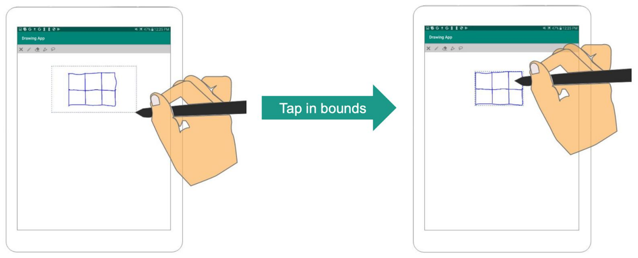

Figure 4: Create a selection, tap in bounds of the dashed rectangle and then move the selected strokes around.

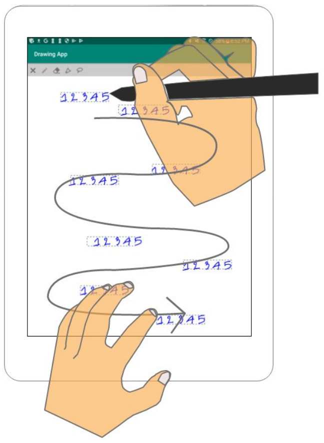

Figure 5. Multitouch copy and paste: the stylus holds onto the selected item and the non-dominant hand index finger drags along the gray line, making copies of the selected item along the trajectory.

Once copies are made and when the user lets go, all objects created will be left selected. This makes it easy for users to take their fingers or the stylus and rearrange each copy individually onto the canvas to their desired location. When the user is done rearranging, they can simply tap onto a not-active portion of the canvas (outside of any bounding box drawn) and deselect all the drawings.

The copy and paste logic involves setting a flag to let the code know that it is in "copy mode". When a user has made a normal selection and a second pointer enters the screen (assuming the first pointer is still within the selected area to keep the selection active), then copy mode begins. Within copy mode, the coordinate of the 2nd pointer is kept track of. This means that all copy and paste functionality is based on the movement of the finger placed down.

The beginning of copy mode starts with creating a GroupCopy object. A GroupCopy object is an object that holds an array of Strokes and a RectF object that defines the bounding box that goes around that array of Strokes. The group of Strokes being copied becomes a GroupCopy object, and this entire object with the Strokes inside is all set to active with a boolean flag called *isActive* set to true. This flag being true means that the bounding box rectangle should be drawn and that the object is movable. The original GroupCopy object is seen as the "stamp" that is getting dragged on the canvas to copy.

As the 2nd pointer moves on the screen, more touch events are created. Every 30 touch events, a new copy is stamped onto the canvas. This copy is made by taking the "stamp" GroupCopy's current location and creating a new GroupCopy object that is a deep copy of the stamp. This copy is left as active and remains in place as more copies are being made.

The only GroupCopy object that is moved is the original stamp. As the 2nd pointer moves on the screen, the displacement is mirrored by all the stamp Strokes and the bounding box RectF object by displacing the coordinates to the pointer's location. To move just the stamp copy, there is a boolean flag listed called *isStamp* that tells the code that when in copy mode, only move the stamp group. Outside of copy mode, normal selection moves Strokes that are flagged with *isActive* as true, so there needed to be a new flag so that copies are left active but not moving.

Once all desired copies are made, the last place the stamp group was left is its final placement, and the GroupCopy's *isStamp* flag is set to false. The stamp is now seen the same way as all copies are seen by the canvas - just a normal active group. Rearranging copied objects is simply done by checking the location of a touch. If the touch is within any bounding box rectangle, then the group will be displaced with the movement of the touch. If a touch is made outside from any bounding box, then all GroupCopy objects and Strokes are set to inactive and normal Select functionality can be done again.

Conclusion

Overall, this application uses existing technology and APIs to conform a note-taking application to the user and their needs. Users are able to utilize the entire screen as their canvas, while being able to switch between popularly used tools naturally with the non-dominant hand that already rests near or on the tablet. Tool changes are faster, and balances the skill needed to operate the radial tool with minimal interruption time from the true task at hand: note-taking.

REFERENCES

1. Apple. 2010. Manipulate objects in Keynote. (2010). Retrieved October 1, 2019 from https://help.apple.com/iwork/1.3/safari/index.html#tan72232abf
2. Benjamin B. Bederson. 2004. Interfaces for staying in the flow. (September 2004). Retrieved September 28, 2019 from https://ubiquity.acm.org/article.cfm?id=1074069
3. Peter Brandl, Clifton Forlines, Daniel Wigdor, Michael Haller, and Chia Shen. 2008. Combining and Measuring the Benefits of Bimanual Pen and Direct-Touch Interaction on Horizontal Interfaces. *Proceedings of the working conference on Advanced visual interfaces - AVI 08 (2008)*. DOI:http://dx.doi.org/10.1145/1385569.1385595
4. GoodNotes. 2019. GoodNotes 5.1 adds gesture control for undo & redo. (May 2019). Retrieved October 1, 2019 from https://medium.goodnotes.com/goodnotes-5-1-adds-gesture-control-for-undo-redo-2dc7db3a87b6

5. GoodNotes. 2019. Introducing GoodNotes 5. (January 2019). Retrieved October 1, 2019 from https://medium.goodnotes.com/introducing-goodnotes-5-387ba7dc3ae0
6. Ken Hinckley, Francois Guimbretiere, Patrick Baudisch, Raman Sarin, Maneesh Agrawala, and Ed Cutrell. 2006. The Springboard: Multiple Modes in One Spring-loaded Control. *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI 06* (2006). DOI:http://dx.doi.org/10.1145/1124772.1124801
7. Ken Hinckley et al. 2010. Pen + touch = new tools. *Proceedings of the 23nd annual ACM symposium on User interface software and technology - UIST 10 (October 2010)*. DOI:http://dx.doi.org/10.1145/1866029.1866036
8. Gordon Kurtenbach and William Buxton. 1991. Issues in combining marking and direct manipulation techniques. *Proceedings of the 4th annual ACM symposium on User interface software and technology - UIST 91 (1991)*. DOI:http://dx.doi.org/10.1145/120782.120797
9. Gordon Kurtenbach and William Buxton. 1994. User Learning and Performance with Marking Menus. *Conference companion on Human factors in computing systems - CHI 94 (1994)*. DOI:http://dx.doi.org/10.1145/259963.260376
10. Shanhong Liu. 2019. Tablets - Statistics & Facts. (May 2019). Retrieved October 1, 2019 from https://www.statista.com/topics/841/tablets/
11. Sachi Mizobuchi and Michiaki Yasumura. 2004. Tapping vs. circling selections on pen-based devices. *Proceedings of the 2004 conference on Human factors in computing systems - CHI 04 (2004)*. DOI:http://dx.doi.org/10.1145/985692.985769
12. Thomas P. Moran, Patrick Chiu, and William Van Melle. 1997. Pen-based interaction techniques for organizing material on an electronic whiteboard. *Proceedings of the 10th annual ACM symposium on User interface software and technology - UIST 97 (1997)*. DOI:http://dx.doi.org/10.1145/263407.263508
13. Ken Pfeuffer, Ken Hinckley, Michel Pahud, and Bill Buxton. 2017. Thumb + Pen Interaction on Tablets. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI 17 (2017)*. DOI:http://dx.doi.org/10.1145/3025453.3025567
14. Margaret Stewart. 2018. 14 of the Best Procreate Gestures that will Save You Time. (April 2018). Retrieved October 1, 2019 from https://theletteringlodge.com/procreate-gestures-save-time/
15. Teoh Yi Chie. 2017. Artist Guide to Active vs Capacitive Styluses. Parka Blogs.Retrieved December 2, 2019 from https://www.parkablogs.com/content/artist-guide-active-vs-capacitive-styluses
16. Yang Zhang et al. 2019. Sensing Posture-Aware Pen+Touch Interaction on Tablets. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Paper 55, 14 pages. DOI: https://doi.org/10.1145/3290605.3300285