LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
Department “Institut für Informatik”
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hußmann

Bachelorarbeit

Single Touch Zoom Gestures on a Mobile Device

Neal Bürger
info@nealbuerger.com

Bearbeitungszeitraum: 01.05.2010 bis 30.09.2010
Betreuer: Dr. Sebastian Boring
Verantw. Hochschullehrer: Prof. Dr. Andreas Butz

Acknowledgments:

First and foremost, I wish to express my gratitude to my advisor, Dr. Sebastian Boring for taking the time to answer all of my questions.

To my parents, for encouraging and supporting me.

To Karla Berkes, for keeping me on track to work on my thesis.

My friends, Fabian Grassl, Anna Munytan, Simon Moissl, Cordelia Wimmer, for listening for hours when I was talking about my thesis.

Abstract:

This project investigated interaction methods on a mobile device that allowed users to hold the device and operate it with the same hand. Four single-touch zoom interaction methods “Tap-Direction”, “Zones”, “Rubbing” and “Circular” were created and implemented on an iOS device, the iPhone 3G. In a user study these methods were compared to the widely known multi-touch gesture “Pinching”. The results show that “Rubbing” is the most inefficient way to zoom, and that “Circular” is a usable, efficient single-touch alternative to “Pinching”

Kurzzusammenfassung:

Das Projekt untersuchte die Interaktionsmöglichkeiten mit mobilen Geräten, die es Benutzern ermöglichen, das Gerät in einer Hand zu halten und es gleichzeitig zu bedienen. Vier ‘single-touch’ Zoom-Interaktionsmethoden wurden entwickelt: “Tap-Direction”, “Zones”, “Rubbing” and “Circular”. Diese wurden auf einem iOS-Gerät, dem iPhone 3G, implementiert. In einer Nutzerstudie wurden diese Methoden mit der bekannten ‘multi-touch’ “Pinching”-Geste verglichen. Die Ergebnisse der Studie haben gezeigt, dass “Rubbing” die ineffizienteste Methode ist zu zoomen, und dass “Circular” die brauchbarste, effizienteste ‘single-touch’ Alternative zu “Pinching” ist.

Tasks:

- Creating Single Touch gestures for zooming on a mobile device
- Design and implementation of the gestures on an iOS device that allows the user to experience the different zoom methods.
- Conducting a user study to determine the most practical input method
- Giving weekly status updates of my work and a final presentation of my work in the "Oberseminar".
- Writing a summary of my work, its underlying concepts and ideas. Its size should be about 50000 words and follow these rules: [http://www.medien.ifi.lmu.de/lehre/bachelorarbeiten/richtlinien.xhtml]

„Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe."

München, den 30.09.2010 _____

Neal Bürger

Inhalt

1. Introduction

Since the introduction of the Apple iPhone many companies followed suit and started to introduce touchscreen-inputs for their mobile device [6]. Mobile touchscreen devices are becoming more and more common in our daily lives. One of the major challenges of mobile handheld devices is that they have to be small so they are limited to a small screen. Even though higher resolution displays built into mobile devices enable a larger workspace for applications, only a limited amount of information is displayed on a screen at any given time.

The most common solution to display more information on the screen is to display the information in full resolution, for example, an image or a map and let the user zoom and pan to the area of interest. It has become a very common task to zoom-in and out to display information.

Even tough zooming has become a very common task, zoom-gestures require usually two fingers. By using two fingers the user often must use both hands to operate the device smoothly. Many users use one hand to hold the device and another hand interacting with the device. This requires the operator to always have both hands occupied while zooming.

Many multi-touch gestures are designed for stationary devices. Typical multi-touch gestures cannot be applied with only using one hand.

a. Motivation

When holding something in one hand it is usually very difficult to operate a device in the other. For a handheld device it should be possible to hold and operate it with only one hand. To test this idea we wanted to create specifically a zoom gesture that could be used efficently in one hand.

We followed the design principle “we design for the extremes, if we understand what the extremes are, the middle takes care of itself”[5]. With a user in mind that is handicapped and is limited to using only one hand, several different interaction concepts were created.

b. The Goal of the Thesis

We wanted to see, if it is possible, to create an efficient user friendly input-method with only utilizing one finger. This would enable a user to only use one hand while operating the device. This way the user has his other hand free and can perform a second task at the same time. Several different approaches were considered and then implemented. Our main criteria was that the interaction method should feel as natural as possible. We then tested our concepts in a user study to figure out which approach is the most promising.

Because zooming has become a very common task, the gestures we considered are only focused on zooming. We did not consider any other functionality with our gestures. The only focus was on how to make zooming with one hand as easy as possible and well accepted by the user.

We found several interaction methods that are well suited. They are futher described in section 3. Single Zoom-inputs.

2. Related Work

Single touch zoom gestures builds on work in specifications of gestures and zoom gestures interactions.

a. Criteria for Interaction Gestures

To create a touch gesture several requirements should be met. The most important aspect is that the gesture is **specifiable** either by example or specification by description. Another point is **Accurate recognition**. It is desired that the user input is evaluated as accurate as possible so the device can immediately react to the input. Hand in hand with this point goes **efficient recognition**. Gestures should be evaluated and recognized as efficient as possible; this in turn allows real-time recognition and immediate feedback. A rather optional requirement is **efficient training**: An ideal system could learn over time how a user inputs gestures and adapt its recognition routines to minimize inaccurate recognitions. This is a very important aspect for handwriting recognition. Another optional requirement is the **device utilisation**. Depending on the sensor frame and other physical factors, each interaction surface has its own characteristics that have to be taken into account [7].

We specified our gestures by description. Accurate and efficient recognition were the most important factors in our design to ensure immediate feedback and accurate zooming. We did not take into account efficient training, because we wanted to create gestures so simple that the user would easily adapt to the gesture. Our gestures were designed to be transferable to any device.

b. Zooming gestures

Fingers lack “resolution” to precisely point at a pixel. A basic technique, called “Take-Off”, is, once the user touches the display that a cursor above the finger tip appears. When the user releases his finger from the display, the object under the cursor is selected/scaled. This enhances the resolution of the finger to 4px [8]. This method needs the object to be correctly selected for scaling the object, so its usage is very limited.

Another technique, “bounding box zoom” or “marquee zoom”, requires that the user first defines a sub-area by drawing a rectangle. A user accomplishes this by first defining a corner and then dragging the finger to the desired opposite corner. Then the device zooms to the contents of the drawn rectangle [4]. The downside is that visual guides and another button to activate the zooming method are needed.

A single-hand technique is “Rubbing in/out”. The user can zoom-in by rubbing from the lower-left-to-upper-right diagonal or zoom-out by rubbing from the lower-right-to-the-upper-left [9]. This interaction method only focuses on how to zoom an image. Alternative interactions, for example, like moving are not considered.

Multitouch gestures, “Expand” and “Contract”, in combination lets a user zoom-in when three fingers pull together on the display, or zoom-out when three fingers on the display move apart [3].

3. Single Zoom-Inputs

We created several different concepts of single-touch zoom gestures that we considered implementing on an Apple iOS device.

To ensure compatibility with common user interactions on such devices, the user can move the image/text with one finger.

We assumed the user holds the device in one hand and operates the device with his thumb, because of the natural way how people hold a mobile device.

a. Interaction Design

In the first design step, we defined the core principals of any zoom gesture:

- Begin-/end-interaction of zooming
- The actual zoom-in/out-gesture
- The positioning of the zoom anchor point, the point to which all scaling is relative to.
- Zoom-in speed needs to be equivalent to the zoom-out speed to ensure a consistent user experience

In addition to these points, we tried to minimize any superfluous visual information. No buttons or any other type of visual feedback should be required. The user should not have to focus on the device while operating the device.

We used an iterative design process to refine and evaluate the most suited concepts to implement. We did not implement all designs, we focused only on the most promising designs. We present some designs that were not implemented to show different aspects that need to be considered for a good design.

b. Concepts

i. Inplace Menu

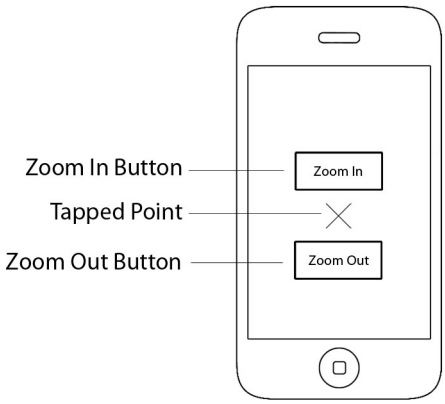

Figure 3.1.: Inplace Menu

The basic idea of an “Inplace Menu“ is to display menu points on the display when needed.

To begin zooming the user needs to tap once on the display with one finger. A button appears north of the point for zooming in and south for zooming out. The user would need to tap one of the buttons to zoom. To end zooming the user would need to either zoom-in/out or

tap anywhere else on the display.

The main advantage of this technique is that it can be easily extended to incorporate several functions besides zooming. In the early design of this method it became clear that several issues needed to be addressed.

When tapping near the border of the display, it is not always possible to display all menu items, for example, tapping near the top edge of the display would result in only displaying the south button. To avoid this issue, the buttons would need to be dynamically placed. But by dynamically placing buttons, the user doesn‘t have a consistent interface anymore.

Another issue is that the user would have to tap exactly on the button to zoom. If the user misses the button, he has to start over again. The buttons would need to be a certain size to avoid false inputs. A disadvantage of buttons is that they obscure the underlying information. By making the buttons larger, it is more difficult to operate the device.

Another issue is that the user is limited to only tapping once on the buttons. This restricts the zooming method to jump in predefined increments to different zoom factors. This could be avoided by allowing the buttons to be pressed longer. During the duration of the button being pressed the user zooms seamlessly. This in turn creates an inconsistent user experience by first tapping then pressing the display.

After considering the pros and cons, “Inplace Menus” were not implemented. The core idea is sound, but, unfortunately, it is not suited for zooming. “Inplace Menus” are a very interesting concept and definitely could be used for other applications.

ii. Scrollbar

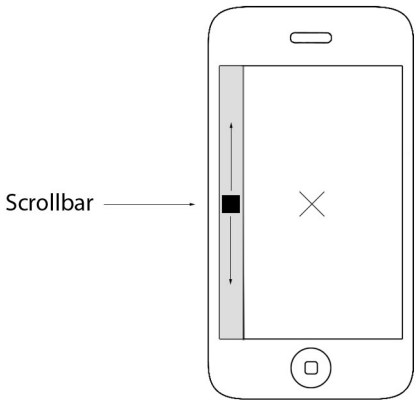

Figure 3.2.: Scrollbar

The user would be operating the display with his thumb. Adding a simple scrollbar to the edge of the display allows the user to visually see at which zoom level currently the image is being displayed. In addition, the user can easily access the scrollbar with his thumb.

The user would be able to zoom by scrolling upwards to zoom-in and scroll downwards to zoom-out. Alternatively by pressing anywhere on the scrollbar, the user could jump to any desired zoom level. No user input is needed to define the zoom anchor point. The point is fixed to the center of the display.

It does not matter on which side of the display the scrollbar is located, a user can interact very easily with a very natural thumb flick to operate the scrollbar.

The downside is that the user must stay within the limits of the scrollbar or the image is

moved. This can be avoided by making the scrollbar relatively wide. By making the scrollbar wide, the scrollbar takes away precious display space. The scrollbar could be translucent, but it would not be visually appealing and still be irritating. Another issue is that users prefer to have as much control over the device as possible. By not being able to set the anchor point, the user does not have sufficient control how the device zooms.

We did not implement the scroll bar interaction method, because the user is very restricted in the way he interacts with the device. Is is also a very desktop based mouse interaction method, more suited to scroll through lists and not very usable for zooming.

Interestingly, the scrollbar concept has been implemented by Nokia in their N97 mobile phone series. All users that participated in our user study that own a N97 stated that they are very frustrated with this type of interaction for zooming.

iii. Press-Zoom

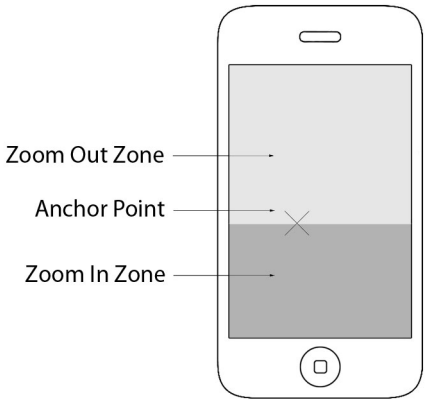

Figure 3.3.: Press-Zoom

We tried to change modes without having the user stop touching the display. Our solution was pressing the display with one finger for a short period of time. The user could move the image and then has to wait for a short time to switch to the zoom mode.

To begin zooming, the user would need to touch the display for around one second. The point pressed would become the anchorpoint. The display would be divided into two zones, all movements north of the anchorpoint result in zooming out and all movements south result in zooming in. Alternatively, the user would press in one of the zones to continue zooming. To exit the mode, the user would need to release his finger.

Unfortunately, we did not figure out how to let the user switch back into the move mode by any kind of pressing; a new gesture needs to be started.

Human perception of a given time period varies greatly between individuals, so finding a proper time interval as required by this method, e.g., one second, may be sensed as too long or too short; users expect a device to react immediately. On top of that, users don't press at the same point for the entire time period. To compensate for that a tolerance area around the pressed point needs to be created.

A draft version of the method was implemented. We ran a couple of tests to figure out a best timing for pressing. But after seeing the high frustration and dissatisfaction of the initial users, we abandoned the further development of this method. Even when the timing was adjusted for a specific user, the user satisfaction did not improve. We concluded that pressing is not a suitable option for switching between modes.

iv. Tap-Direction

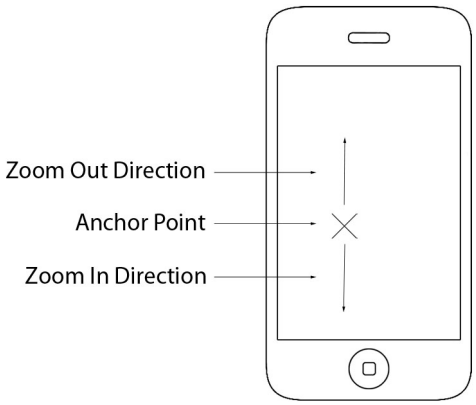

Figure 3.4.: Tap Direction

Using a single tap on the display enters zoom mode. In zoom mode, the direction of movement defines the zoom type: moving upwards zooms in, moving downwards zooms out.

To start zooming the user has to tap the display, this place serves as anchorpoint. To zoom-in the user moves his finger upwards. But if he zooms too far, he can simply correct it by moving his finger downward. To exit zoom mode, the user simply releases his finger.

This method is very simple. Its main advantage is that one can immediately change from zooming in to zooming out. It also can be operated with a thumb flick. The user is free to move in any way he likes, as long as he continues in the same direction, the zoom method stays the same.

The interaction is limited to the display. If the user has not finished zooming or starts the interaction too close to an edge, the user runs out of space to complete the interaction. He must start another gesture to continue zooming.

v. Zones

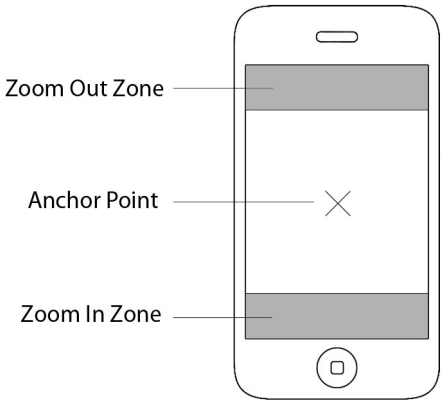

Figure 3.5.: Zones

Having to first enter a zoom mode, via tapping or pressing, can be annoying. An alternative method was investigated to see if the mode switching problem could be avoided from the start. “Zones” divide the screen into three input areas, invisible to the user. The top area of the screen allows the user to zoom-out, the bottom area allow the user to zoom-in, and the area in the middle allows the user to move the image.

Since the application is already in the zoom mode, the user simply starts zooming by touching one of the zooming zones. The user can cross the edges of the zones, but still stays in the same interaction method he selected by touching the screen. Since there is no way of figuring out where exactly the user wants to zoom to, the anchorpoint is simply set in the center of the display.

This "Zones" mode can easily operated with a thumb flick. The user can switch quickly between modes because no tapping/pressing is required. Once the user has defined an operating mode, he is not restricted to any specific movement. He can move in any direction and still the same zoom method is applied.

The major downside of this method is that the user cannot define the anchorpoint. To zoom to the area of interest the user must always first center that region on the display. But the user does not know where exactly the center of the display is, so he always must approximate and does not get the desired results.

vi. Rubbing

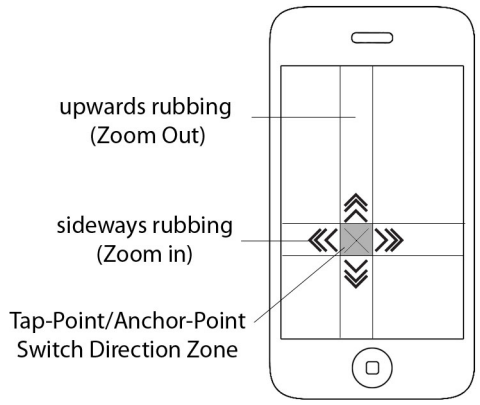

Figure 3.6.: Rubbing

As with most gestures, if you start too close to the edge, you must restart the complete gesture because you run out of space. Rubbing allows the user to extend the length of interaction by allowing the user to reverse the direction of movement, but continues to interact with the device in the same manner as before. To further enhance this user experience, switching between modes is seamlessly possible.

The user would start by tapping the display, thus defining the anchorpoint and then rubbing sideways to zoom-in or rubbing upwards to zoom-out. At the anchorpoint the user could switch rubbing directions. To exit the zoom mode the user simply releases his finger.

Overall this gesture is very well suited for zooming. It addresses all problems encountered with other gestures. Even tough the user is restricted to either a sideway or a upwards movement, rubbing itself is a very natural gesture and is easy to operate for any user.

The only remaining issue is how to handle the switching between the "move" interaction and "zoom" interaction. The solution is to tap in the beginning of the gesture. Without tapping, the initial rubbing gesture would move the image around and irritate the user.

vii. Circular

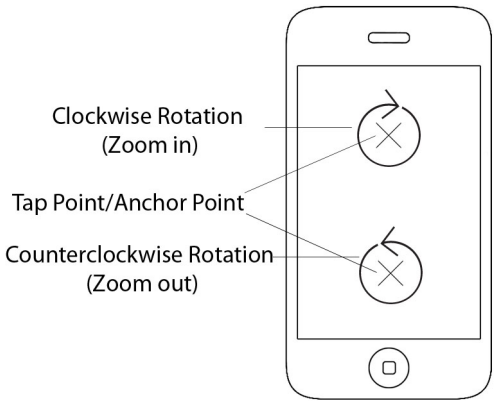

Figure 3.7.: Circular

Many physical devices use knobs and dials to control, for instance, the volume. Users know how to use dials to increase or decrease a value. The circular motion was partially based on how a screw operates, screwing the screw into the wall you need to rotate the screw clockwise, to screw outwards you need to rotate counterclockwise.

The gesture starts with tapping the display to define the anchorpoint. The user would then circle clockwise around the anchorpoint to zoom-in (counterclockwise zooms out). If the user makes an error, he simply reverses the direction. To end the gesture, the user simply stops touching the device.

A circular motion is clearly differentiated from the move gestures that users associate with moving an image (up, down, left, right). In addition the circle could be continued indefinitely thus avoiding the limitations of the screen size and one can immediately switch from zooming in to zooming out by simply reversing the direction. Drawing a circle with a thumb on a display is also a very easy and natural movement.

c. Implementations

The interaction gestures were implemented in Objective C for iOS Version 4.0. The touch device used later was an iPhone 3G.

i. Zoom to Point

All interaction gestures would call the same zoom function to handle the user input. In iOS the anchorpoint is always set to the center of the display. In the first step the anchorpoint had to be set to a new position on the display. This results in a translation of the image. The image then needs to be translated back to the original position by the difference of the old anchorpoint and the new anchorpoint. Then the image can be scaled accordingly.

It is important to note that the user input position has to be converted into image coordinates and then into normalized display coordinates. If the input position would be directly converted you cannot compensate for previous translations and scaling of the image.

ii. Tap Direction

The implementation of “Tap Direction” is rather straightforward.

Once the display is tapped, the anchorpoint is defined as a position on the image.

While the user moves his finger on the display, the y-coordinates of the current position is subtracted from the y-position of the last position. On iPhone devices, the origin of the screen coordinates is located on the top left corner. If the user is moving upward, the difference between the two points is positive (zoom-in), moving downward it is negative (zoom-out).

In the rare case that the user moves sideways, nothing happens until again a difference between the two points can be detected.

Having "Tap Direction" implemented with "upward/downward" being rather loosely defined, allows the user a very high range of freedom how he can operate the device without errors.

iii. Zones

In our implementation of "Zones" we divided the screen into two equally sized areas of 20% for zooming in/out and the remaining 60% of the screen for moving the image. On an iPhone 20% of the screen height is 1.5 cm slightly smaller than an average adult finger (2cm), this works fine because the touch occurs with the tip of the finger.

The anchorpoint is defined by the center of the display with the XY-coordinates (`display width /2, display height/2`). The coordinates are then projected into the image coordinate system and adjusted to compensate for any scaling and/or moving that has previously occurred.

Once the display is tapped, the device analyses the touch position. The zoom mode variable is set to 0. If the touch y-position is in the `display height * zones width`, the zoom mode variable will be 1. If the touch y-position is in the `display height * (100 - zones width)`, the zoom mode variable will be 2.

While the finger is moving on the display, depending on the zoom mode, the device will react as follows: move (zoom mode = 0), zoom-in (zoom mode = 1), zoom-out (zoom mode = 2).

iv. Rubbing

The initial tap position serves as the anchorpoint and defines 5 interaction zones:

The switching zone - it is directly positioned at the anchorpoint serving as center with a width and height of 40px approximately 0.8cm.

Two zones for zooming out - they are located above and below the switching zone. The width is 40px and the combined height of both zones and the switching zone equals the display height.

Two zones for zooming in - they are located left and right of the switching zone. The height is 40px and the combined width of both zones and the switching zone equals the display height.

This may result in one/two zones having a height or width with value 0px. To ensure that the user is actually rubbing, the movement must pass over the switching zone three times.

After the initialization of the rubbing mode, as long as the touches are in one of the four zooming zones, zooming occurs according to the active zone.

v. Circular

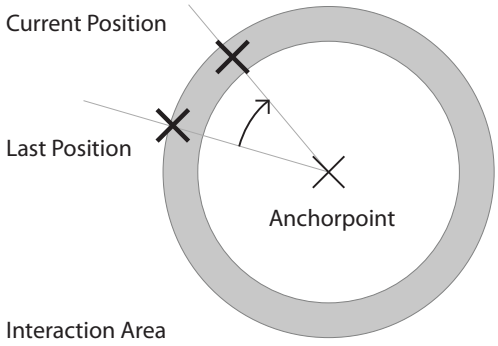

Figure 3.8.: Mathematical Approach

In the first attempt, we assumed that a circular user input would be based on a mathematical circle. Two circles with different radius were created around the anchorpoint. In the area in between the circles the gesture would be recognized. A straight line was created between the last position and the anchorpoint, a second line was created between the current position and the anchorpoint. Depending on the angle between the lines the rotation direction was determined.

This approach turned out to be totally flawed. We discovered that a human without visual guides usually does not even come closely to draw a circle. Further, users also have difficulties to draw an ellipse-like shape. Users tend to start at one point and end somewhere totally different.

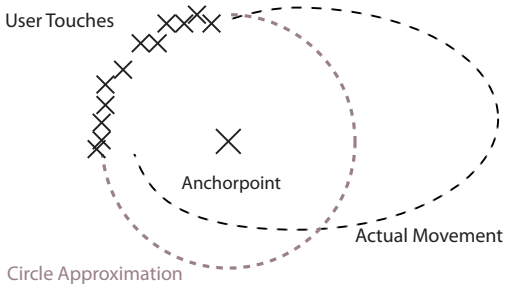

Figure 3.9.: Movement analysis approach

The gesture would need to find a very “loosely” defined circle. In the second attempt, we tried to analyze the movements of the user on the device. We hoped to find “round” movements by comparing how the current position relates to the past positions. This created memory issues and the response time of the device was unacceptable, because the user first had to draw a quarter of a circle until the device reacted and often the user input could not be properly analyzed. In many cases the user draws for short periods a straight line. The algorithm could not properly handle those cases.

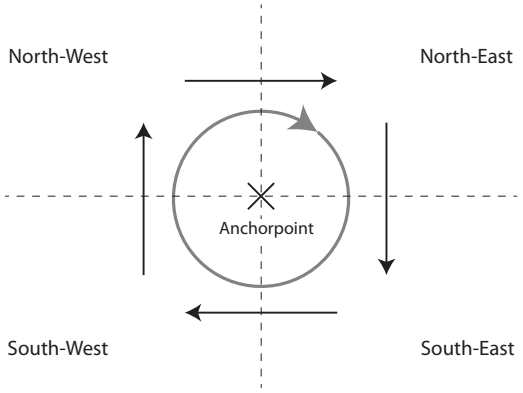

Figure 3.10.: Directional Movements Approach

In the final attempt, we drastically simplified the problem. Starting from a square, we derived that to draw a circle two basic movement types are required, an up/down movement and a left/right movement. We split the square into North/South and West/East. So, for example, in the NW area one needs to draw a line upwards and then right to draw the square. The same movements are required for drawing a circular shape. To draw a circle clockwise, one needs to draw in the NW, up, right, NE, left, down, SE, down, left, SW, left, up. (For counterclockwise NW, left, down, SW, down, right, SE, right, up, NE, up, left)

The core principle of the algorithm works as follows:

First the user taps the display, defining the anchorpoint. The anchorpoint serves at the same time as the division point of the display. All points above the anchorpoint are North, below are South, to the left are West and to the right are East.

While the user is moving his finger, the device compares the current position with the last position, determining the two direction types up/down and left/right. It then checks for the area, if the two conditions (up/down, left/right) of the movement are correct, for example, in the area NW, up and right, then equals a clockwise movement.

Straight lines parallel to the horizontal or vertical axis are not recognized by this algorithm. However, the user draws straight lines for only very short periods. As soon as the user makes correct inputs again, the device continues to zoom without noticeable delay effects. Since the algorithm only checks the directions, one can draw any round shape in any size around the anchor point, and the algorithm would be able to handle the input. In essence, you can spiral outwards from the beginning and the input would still be valid.

4. User-Study

We conducted a user study to figure out which of the different interaction methods would work best. The remainder of this text is structured as follows: In the first part, the conditions, tasks, study design, participants, and hypotheses are descibed, in the second part, study results, qualitative feedback, and observations are presented. The last part has a discussion of the results.

a. Conditions

The “Tap-Direction” condition allowed users to interact with the device with moving the image with one finger, zooming out with tapping and moving the finger up, zooming in with zooming in with an initial tap and then moving the finger downwards.

The “Zones” condition allowed users to zoom-out with the upper zone, move the image with the middle zone and zoom-in with the lower zone.

The “Rubbing” condition allowed users to move the image with one finger, zoom-in via inital tap and sideways rubbing and zoom-out via inital tap and up/down rubbing.

The “Circular” condition allowed users to move the image with one finger, and after inital tap any clockwise motion to zoom-in and counterclockwise motion to zoom-out.

In addition we used “Pinching” as a baseline condition. “Pinching” is a multi-touch gesture that cannot be efficiently used with only one hand. Users usually place the mobile device in one hand and operate it with the other hand.

The “Pinching” gesture works as follows: by placing two fingers on the screen and spreading them farther apart the user zooms out, or by bringing the fingers closer together the user zooms in. The anchorpoint is defined as the point between the two fingers.

The users could not switch between the interaction modes during testing. Users could operate and hold the device any way they wanted except for “Pinching” where the user was required to operate the device with two hands.

b. Tasks

The participants had to perform the task of finding and filling the screen with a red square on a city map. Depending on the task the initial size of the square had a initial size of 140px for zooming in or 640px for zooming out.

To start the task the participant created a square by double tapping with two fingers on the display. The display was then immediately reset to the initial state with a zoomfactor of 1.0 and then the user had to locate the square. To end the task the participant had to place the square on the display in such way that the square width was equal to the display width and all four corners of the square were visible. The user had a tolerance of 40px, but the square could not be bigger than the screen. If the participant was successful, he could remove the square by double tapping with two fingers on the display.

The two finger double tapping gesture was chosen because it did not conflict with any other gesture tested in this study and was consistent throughout with every interaction method.

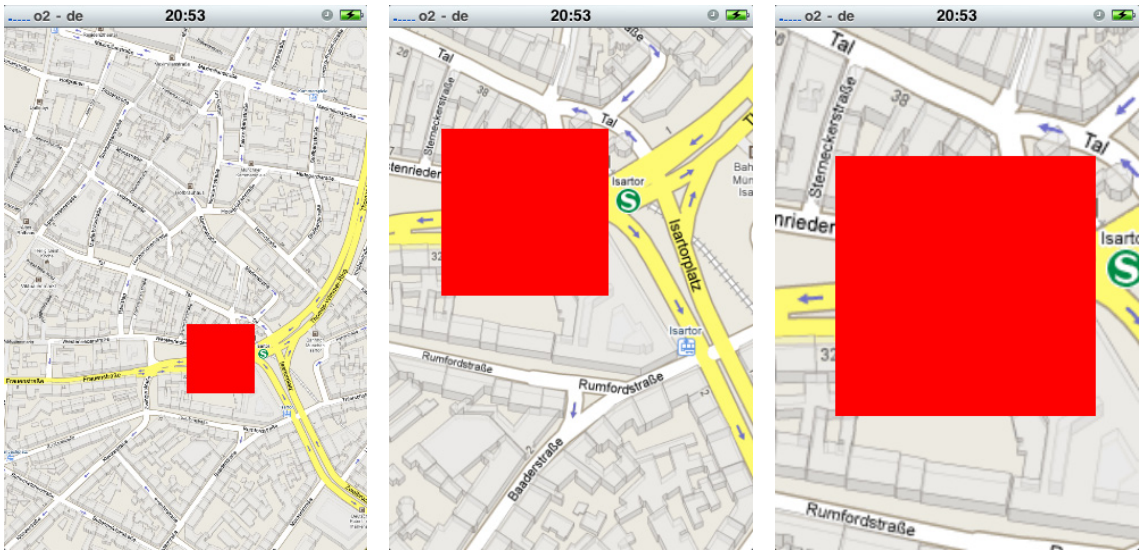

Figure 4.1.: Figure: from left to right: 1) the display after creation of a small square. 2) zoomed in but the square is not large enough for removal. 3) the square large is large enough for removal.

c. Study Design

The conditions were permuted for the entire study with the "latin-square" technique.[10]

An interaction method was presented to the participant and then the device was given to the participant. The participant received three test squares he had to create and remove before the actual study commenced.

In the actual study the user was given at random eighteen squares. The squares were located at nine different locations located NW, N, NE, W, Center, E, SW, S, SE of the initial display location.

The study was then repeated with the same participant for every single interaction method.

Even though the gestures are designed for single-handed interactions, users were permitted to use both hands. This ensured that the total duration of the user study was under 60min.

d. Participants

Fifteen participants (10 female, 5 male, aged 21 to 26) volunteered for the study. The participants were students from various fields, for example, medicine, chemical engineering, art and multimedia. Only two participants were active iPhone users and four users were familiar with touch based devices.

e. Hypotheses

We hypothesized that "Zones" would be the slowest interaction method of the five conditions, because of the fact that the users cannot set the anchor point as needed.

For "Pinching" we assumed that it is the most efficient way to zoom compared to the other conditions, as users, even having never come in contact with a touch device, had seen this gesture in action due to advertisements from Apple [1]. "Pinching" has been implemented as default zoom-interaction on the iPhone and it is a very simple zoom gesture.

In contrast, we assumed that “Rubbing” is the most inefficient way to zoom compared to other gestures. “Rubbing” seems a rather unnatural gesture for zooming; users cannot associate rubbing with zooming.

f. Study Results

We compared separate repeated measures using analysis of variance (ANOVA)[10] tests on mean completion times and errors for each task. For the zoom-in as well as the zoom-out task, the duration was measured from creation to removal of a square. We measured the margin of error by measuring the offset defined by comparing the removed square with the actual size of the square. The square could be up to 40px smaller than the original size and still be removed.

The input data for the ANOVA method were the interaction method (Technique), square size (Size), with deviation and offset as measured values.

Duration measurements: We found a significant main effect on the task completion time of Technique ($F_{4,56}=33{,}989$, $p<0.001$), only “Rubbing” and “Zones” differ from “Pinching” with ($p<0.001$ and $p<0.12$), meaning that “Rubbing” and “Zones” need significantly more time to complete a task than “Pinching”. At the same time it means that “Circular” and “Tap-Direction” do not differ from “Pinching”. We further found an interaction between Technique and Size ($F_{4,56}=4.386$, $p<0.05$). “Rubbing” vs “Pinching” and “small size” vs “large size” $p<0.24$, and “Zones” vs. “Pinching” and “small size” vs. “large size” $p<0.12$, with other methods having no significant effect. This means that it takes longer to zoom-in than to zoom-out and that the interaction method is not symmetrical. But this measure may be distorted because users first zoomed out to find the square and then zoomed in.

Offset measurements: We found no significant main effect on the offset for the Technique ($F_{4,56}=0.675$, $p<0.46$). Apparently all gestures had similar accuracy, or the tolerance value was set too low to find any significant differences between the results.

The test results of “Tap-Zoom” were consistent with all test subjects. The test results had minor fluctuations. No users showed any learning curve or fatigue. (The mean duration of all test results was 12.1s.)

During the testing of “Zones” five participants showed signs of fatigue after completing the first six zooming tasks. The average duration of the first six squares was 12.2s and the last six squares had an average of 21.4s. Six participants had steady test results. The remaining four participants had a minimal improvement over time from an average of 19.6s in the beginning six squares to 14.2s in the last six squares.(The mean duration of all test results was 16.6s.)

While working with “Rubbing” seven participants required more than 60s to complete a single zooming task on several occasions. Four of these users needed longer than 120s to complete a single zooming task. A single user showed a minor learning effect, improving his speeds from 60s to 22s on average. All remaining users had high fluctuations in their test results, ranging from 15s to 40s. From the test data we could not determine if learning effects or fatigue was taking place. (The mean duration of all test results was 28.8s.)

Seven participants showed a very rapid learning curve while using “Circular”. The first six squares had an average duration of 20.1s, and the last six squares had an average of 8.5s. The other eight participants had a consistent average speed of 9.5s per square. The average improvement of all participants was 4.5sec. No user showed signs of fatigue. (The mean duration of all test results was 13.1s.)

All test subjects showed consistent test results when using “Pinching”. The users had no significant fluctuations. No users showed any learning curve or fatigue. (The mean duration of all test results was 10.5s.)

g. Qualitative Feedback

After finishing the trials for a interaction technique, the participants filled in a questionnaire in which they ranked the experienced technique on a scale from 1 to 5 with regard to smoothness of operation, accurate zooming, perceived operation speed, finger fatigue, wrist fatigue, arm fatigue, shoulder fatigue, general comfort, and the ease of the overall input method. They were also encouraged to provide written and verbal comments. The experimenter also took notes of interest that occurred during the test. At the end of the study, they commented on the techniques they liked the most and the least. A summary of the results is shown in Figure 4.3.

Smoothness of operation

For “Rubbing” most participants could not stay in the bounds of the rubbing zones resulting in unexpected responses from the device. Many participants stated that “Rubbing” is “completely useless”.

“Zones” was poorly received as well, mostly because the user could not set the anchorpoint and was always required to zoom to the center which resulted in undesired zooming and additional panning.

“Tap-Direction”, “Circular” and “Pinching” were all equally well received.

Accurate zooming

Users had the most difficulties with “Rubbing”. While rubbing, many participants forgot where they actually tapped and where the anchorpoint was set, so they had to restart the gesture.

On top of that the most common error with “Zones” was that the users started at the exact top of the display. This area is usually used as a menu bar position, so in many cases the iPhone did not register the initial touch as part of the gesture.

Figure 4.2.: User touches the menubar which results in an faulty input

“Tap-Direction”, “Circular” and “Pinching” were all very well received.

Perceived operation speed

"Tap-Direction", "Zones", "Circular" and "Pinching" are perceived as having optimal operation speed. Even though the zoomspeed was set equal for all interaction methods, "Rubbing" was perceived as too fast. Users stated that they could not judge when the zooming started and due to the rapid movements of the finger to rub, the device zoomed very fast.

Fatigue

All interaction methods had a slight finger/wrist fatigue, again with the exception of rubbing that had a very high finger and wrist fatigue. Hardly any participant had any arm or shoulder fatigue.

General comfort

Users were most uncomfortable using "Rubbing" or "Zones". The participants were very comfortable using the remaining interaction methods.

Overall input method

The participants liked "Tap-Direction", "Circular" and "Pinching" equally well. "Zones" was neither liked nor disliked. Only "Rubbing" was disliked. One participant commented "simply everything is wrong with this gesture".

Most preferred gesture

Participants were asked to comment and specify the techniques they liked the most. All participants preferred either the "Tap-Direction" (7 users), or "Circular" (8 users). Interestingly even the iPhone users preferred "Tap-Direction" over "Pinching".

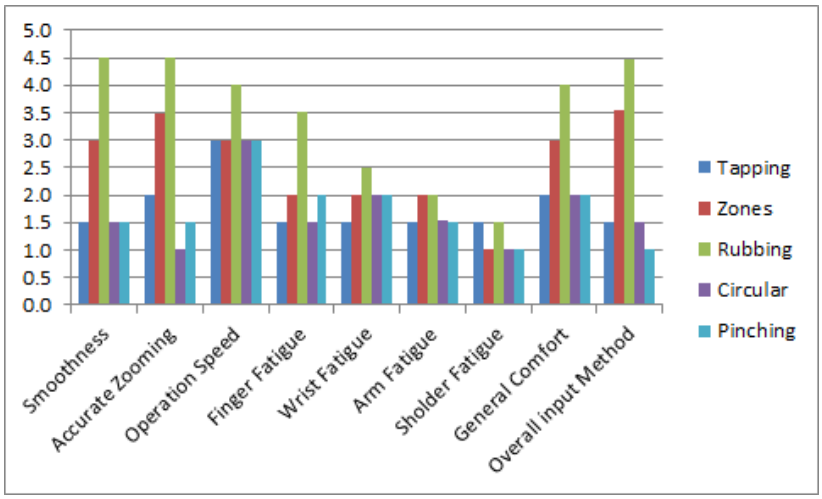

Figure 4.3.: Qualitative Feedback summery, lower values are better

h. Observations

i. Zooming

All users first zoomed out to get an overview over the complete image and then zoomed in into the image to focus on the square. No user panned to find the square.

The users were very satisfied when they could complete the zooming task in around 10s, values above 20s resulted in frustration and dissatisfaction.

ii. iPhone User

One of the participants uses his iPhone for around 0.5-1.0 hours a day, playing several iPhone games. With the exception of rubbing, this user was equally fast, with either method presented. This indicates that all interaction methods could be learned and applied efficiently.

iii. Fingernails

Several women that were using an iPhone could not operate the phone properly because of long fingernails. This lets this user group prefer other display types which they can operate with fingernails.

iv. Tapping

Seven participants had difficulties tapping on the iPhone. The device only recognizes taps that are at the exact same point with very little tolerance. The participants tapped roughly 1cm next to the first touch on the screen, resulting that the device couldn't handle the input.

After several similar observations it became clear what the users were trying to do. The users wanted to tap on the object of interest and then control the zooming by always having a direct visual feedback without ever obscuring the object with their finger. This lead users never to tap exactly again on the object, thus the device did not register a tap.

To enable these users to tap more efficently a radius zone around the first tap needs to be applied.

v. Rubbing

Users are very irritated with the nature of rubbing. They do not understand why they have to move their finger two times back and forth. Users expect when they change direction that the reverse should happen. This is equally valid for "Rubbing" and for "Zones", where rubbing is also possible. It does not make sense to use any form of rubbing for a zoom gesture on a mobile device. Previous studies have shown that rubbing with increments is possible.[9] Our study has shown that seamless zooming with rubbing on the image does not make sense. In addition, as previous studies have shown, rubbing causes high friction with the finger and creates an unpleasant experience.

i. Comparison

When comparing the offset and duration values, "Circular" has the highest accuracy and "Pinching" has the highest speed.

"Circular" and "Tap-Direction" have very similar results towards the baseline criteria "Pinching".

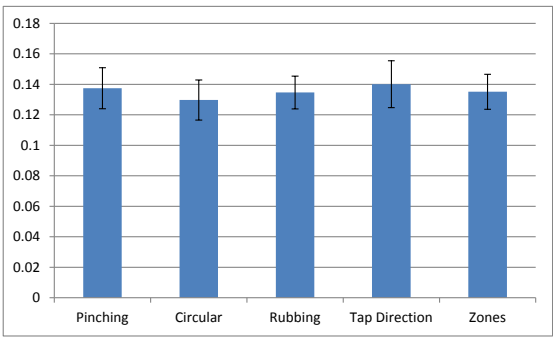

Figure 4.4.: Mean of all offsets

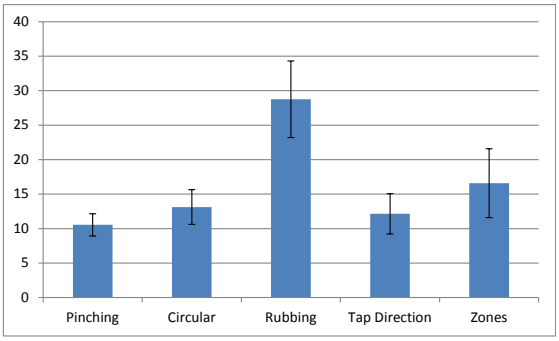

Figure 4.5.: Mean of all Duration Values

j. Discussion

The study shows that "zooming in" is slower than "zooming out" in all interaction techniques. This is a direct result of the participants zooming out to get an overview and then zooming in.

The participants focused mainly on completing the tasks as fast as possible, this resulted that the users stopped focusing on accuracy and only tried to be as quick as possible. The offsets do not show any statistical relevant data.

"Rubbing" is the slowest interaction method with a mean of 28.7s. This interaction method was not perceived well. Users were frustrated with it, it took too long for completing the tasks. The high fluctuation of the test results indicate that not a single user could properly operate the device as intended.

"Tap-Direction" is the fastest interaction method with an mean of 12.1s, but at the same time, it is the interaction method with the highest offset of 0.14%. Instead of restarting the gesture when the user reaches the end of the screen, he simply tries to remove the square to complete the task, thus resulting in a faster speed but higher inaccuracy.

The very rapid learning curve of the "Circular" interaction indicates a trend that in the long term usage the average operating speed would be around 8.5s.

5. Summary

The project investigated interaction methods with a mobile device requiring only one hand to operate. Four interaction methods were implemented and tested for their usability. They were also compared to “Pinching” that has been introduced by Apple.

a. Comments

“Tap-Direction” or “Circular” would be good interaction methods for single handed zooming. Even though “Circular” is initially not the fastest interaction method, users tend to work with the gesture more precisely. Its very rapid learning curve lets users work efficiently with only repeating the gesture a few times. “Circular” is extremely well suited for zooming with only one touch.

It is very interesting that “Tap-Direction” and “Circular” have similar results as “Pinching”, especially as “Pinching” cannot be used by using only one hand. This suggests that these alternatives to “Pinching” would work well in all usage scenarios.

b. Future Work

We only explored the possibility of efficient zooming with our gestures. It would be very interesting to further explore the possibility of extending a gesture to multiple functionality. Also the gestures could be enhanced by the possibility of using visual guides.

In an alternative version of “Circular”, the user would have the ability to visualize four different interaction methods in the different zones. By pressing in one of the zones, the user could activate a different function, for example, use the main gesture as volume control, pressing in the NW the song plays/pauses, pressing NE goes to the next song, pressing SE goes to the previous song and pressing SW stops the playback.

This interaction method could be appilied also in other enviroments, for example, in an automobile to control the radio.

When using visual guides, tapping could be replaced by some other sensor input than from a touch display. By utilizing the motion sensors or a digital compass [2] other types of switching between modes becomes possible, avoiding the use of tapping on the display.

6. References

[1] Apple Computers Inc., “iPad introduction”, commercial, http://www.apple.com/ipad/, U.S.A., 2010

[2] Apple Computers Inc., iPhone, http://www.apple.com/iphone/. Sep. 2010

[3] Apple Computers Inc., “Multi touch gesture dictonary”, Patent US20070177803, 2007

[4] Albinsson P., Zhai S., “High Precision Touch Screen Interaction”, Proceedings of CHI 2003, pp. 105-112

[5] Gary Hustwit “Objectified” Dan Fermosa Interview, 2009

[6] Research in Motion Press Release, “AT&T and Research In Motion Ignite Customers with the New BlackBerry Torch” http://press.rim.com/release.jsp?id=4238 August 3, 2010 accessed September 28, 2010

[7] Rubine, D.H., “The Automatic Recognition of Gestures,” CMU-CS-91-202, Submitted in Partial Fulfillment of the Requirements of the Degree of Doctor of Philosophy in Computer Science at Carnegie Mellon University, 1991 p.18-21

[8] Sears, A. and B.Shneiderman, High Precision Touchscreens: Design Strategies and Comparision with a Mouse. International Journal of Man-Machine Studies, 1991. 43(4): p. 593-613

[9] Olwal, A., Feiner, S., and Heyman, S. “Rubbing and Tapping for Precise and Rapid Selection on Touch-Screen Displays”. Proceedings of CHI 2008 (SIGCHI Conference on Human Factors in Computing Systems), Florence, Italy, April 5-10, 2008, pp. 295-304.

[10] C. F. Jeff Wu, Michael S. Hamada, “Experiments: Planning, Analysis, and Optimization , 2nd Edition”, ISBN: 978-0-471-69946-0, Wiley 2009

7. Contents of attached CD

/src

/references

- References available on the available in Adobe PDF-format.

Numbered according to the “References” list

/documentation

/presentation

SingleTouchZoomGestures.pdf (this document as pdf-file)