RWTH AACHEN UNIVERSITY

Touch Graffle:
Developing a Diagramming Software for Large Multi-touch Tables

Master's Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Claude Bemtgen

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Johannes Schöning

Registration date: 11.03.2015
Submission date: 31.03.2015

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, March 2015
Claude Bemtgen

Contents

List of Figures

List of Tables

Listings

Abstract

Diagrams are a popular resource to convey the relationship between different objects or states. Tools that let people create these kinds of graphs on a standard computer are numerous and everyone has its own advantages and disadvantages. However, they have one thing in common; they have all been designed to be operated with a single mouse and a keyboard. With the growth of the tablet market, enterprises started gradually porting their products to these mobile devices. Even though they encourage the use of multiple direct input points, the implementations are only copies of the desktop versions that do not take into account the higher degrees of freedom.

Touch Graffle addresses this issue. It does not only run on a larger, more overseeable multi-touch table, but it actually exploits the advantages of multiple, simultaneous input traces. I imagined an attuned gesture set, that allows a facilitated and straightforward multi-touch manipulation of diagrams. Like this, the user can focus on her actual task and does not need to think about how to reach her goal. As a benefit, I will also compare the different strategies that users pursue on the various surfaces.

Überblick

Wenn es darum geht Relationen zwischen Objekten grafisch darzustellen, sind Diagramme genau das richtige Hilfsmittel. Computerprogramme, welche das Erstellen und Editieren solcher Art von Graphen ermöglicht, gibt es wie Sand am Meer und alle haben ihre eigenen Vor- und Nachteile. Was sie allerdings alle gemeinsam haben, ist dass sie entworfen wurden um mit einem einzigen Mauszeiger und der Hilfe der Eingabetastatur bedient werden zu können. Als der Markt der Tablets anfing zu wachsen, wurden viele dieser Programme auf die mobilen Geräte übertragen. Doch obwohl Tablets die Benutzung von mehreren gleichzeitigen Eingabepunkten fördern, sind die meisten Applikationen nur Kopien von den Desktop Versionen und somit gar nicht auf die direkte Manipulation abgestimmt.

Touch Graffle geht nun einen Schritt weiter, indem es nicht nur auf einem größeren und übersichtlicheren Multi-Touch-Screen läuft, sondern die Vorteile der mehrfachen Eingabepunkte ausnutzt. In meiner eigenen Umsetzung habe ich mir einen abgestimmten Gestensatz überlegt, welcher die Handhabung von Diagrammen vereinfacht und intuitiver gestaltet. Somit muss die Benutzerin nicht darüber nachdenken wie sie was editieren kann, sondern kann darauf losarbeiten und sich auf ihre eigentliche Aufgabe konzentrieren. Als weiteren Gewinn werde ich die verschiedenen Strategien vergleichen, welche die Nutzer auf den unterschiedlichen Geräten verfolgen.

Acknowledgements

First of all, I would like to thank Prof. Dr. Jan Borchers and especially my supervisor Simon Völker for giving me the opportunity to work on this interesting project and for backing me with feedback and helpful advices during the whole venture.

Furthermore, I am thankful for the pleasant atmosphere on the Media Computing Group chair and special thanks go to the ten participants who consented to help me out during my two studies.

Last but not least, I really appreciate the support of my parents, who always stood by me during my whole studies . Thank you!

Conventions

Throughout this thesis I use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

`myClass`

The whole thesis is written in American English.

Chapter 1

Introduction

A graph is worth a thousand words — It might not be the right proverb, but there is definitely some truth behind it. Instead of using a textual form to explain certain relationships between different variables, one can also create a graphical representation to illustrate these relations. In most cases this needs less text, is clearer and can be grasped in a shorter time than its textual match. No matter what type they are, if it is a bar chart, a state diagram, a pie chart or any other kind of graph, they always have one main goal: visually show the relationship between different variables.

Graphs show the relationship between variables

On normal desktop computers, there exist a bunch of applications which let you create all kinds of graphs. Microsoft's Excel[1] for example lets you plot your charts based on the numerical data you provide. Omnigraffle[2], an application for OS X or the iPad, or Lucidchart[3] for Microsoft Windows, are only two examples of programs that let you create all kinds of diagrams from scratch. These were only some of the most famous ones, but the list of different applications is long, and each of them has its own subgroup of graphs it can create, its own advantages and disadvantages and also its own interface.

On the desktop computer, there exist many tools to create graphs

[1] https://products.office.com/de-de/excel
[2] https://www.omnigroup.com
[3] https://www.lucidchart.com

Multi-touch becomes more and more popular

However, what about touchable interfaces, especially interfaces that support multi-touch? Since the introduction of the iPhone in 2007, life without direct-touch interfaces is almost inconceivable. But not only smartphones, which often only need to recognize one or two points of contact on the surface, also bigger surfaces have become more and more widespread. These surfaces, with the dimensions of tables, have the advantage that they are simple to use and of an intuitive nature. On these screens, the users can theoretically use an infinite number of fingers as an input source. Therefore, these systems are very popular as collaborative workstations, where multiple people can take place around the desk and operate it simultaneously. In general, one can say that multi-touch lets the users work more efficiently and effectively as they focus on the task at hand, rather than the technology behind the system.

We need to define an expressive, easy and non-overlapping gesture set

Due to the necessity of graphs and due to the propagation of multi-touch surfaces, the combination of both should be studied in more detail. Particularly a set of gestures needs to be defined, where the different gestures are meaningful, easy to remember and do not overlap. There is already a certain number of gesture-function-pairs that are being used without any reflective thinking, like the two-finger pinch gesture, however for more than two fingers there is still a lot of leeway and every application can make use of them in the best possible form.

The main contribution of this thesis is the creation of a system, combined with the assembly of an attuned multi-touch gesture set, that supports the individual in the creation of graphs on tabletops. Speed, efficiency and intuitiveness are the central keywords that are being taken care of. One benefit is also the negligence of modes or tools. Depending on the gesture, the system knows what mode the user is in, so that we could speak of *dynamic tool switching*.

The following small explanations of the different chapters will elucidate the structure of this document and will give an overview on what can be expected.

The second chapter contains the related work part, where I present research that has already been done in the field

of multi-touch and tabletops, and which has influenced my design, my gesture set and the overall interaction with the system.

In the following chapter, I will focus on graphs. After some history of how charts have been invented, I will answer the question on what the most important aspects of graphs are, in order to know what users absolutely need so they can transfer their mental model onto the screen.

In the fourth chapter, I will explain the procedure of my pilot study, which helped me to find out how users intend to operate applications that let you create graphs. Besides, I will try to find sequences that could be accelerated using the power of multiple fingers and not only one mouse cursor respectively one single touch. In my user study, I asked my participants to recreate a given graph on a computer and on a tablet.

The fifth chapter will cover the design and implementation of my own work. Based on the things I figured out in the study, I will explain my user interface and my gesture set. Both will try to optimize the creation of graphs on a multi-touch surface.

After that comes the evaluation chapter. In this one, I will present my software to the same users that helped me out in the fourth chapter, and ask them to recreate the given graphical illustrations again, but this time on the tabletop. By comparing the results to the results I could gather before, I can grade my work and see where there are still some flaws.

In the seventh chapter, I will sum up the whole thesis and point out these previously mentioned remaining flaws. I will try to identify the origin of these problems and suggest potential solutions.

Chapter 2

Related work

There were many other publications that influenced my work and that helped me make some design decisions. When reading through the literature, I had a few categories in mind that were of interest to me. Naturally, at the beginning, I was looking for similar topics to see how other authors imagined the process of object manipulating applications. Furthermore, I was looking for possibilities to present an interface to the user that lets her edit the properties of the graphical objects. On desktop computers, this is mostly done with menus, but what about touchscreens? And especially about large tabletop surfaces? Last but not least, as touchscreen surfaces not only allow, but even *assume* multiple fingers to be used during the action, existing multi-touch gestures were also of interest to me.

Important categories for me are other similar tools, menus and existing gesture sets

So all in all, in this chapter I will consider four main fields of research in the following order: ”Large direct-manipulation interfaces”, ”Multi-touch gestures”, ”Menus on multi-touch surfaces” and ”Object manipulating applications”.

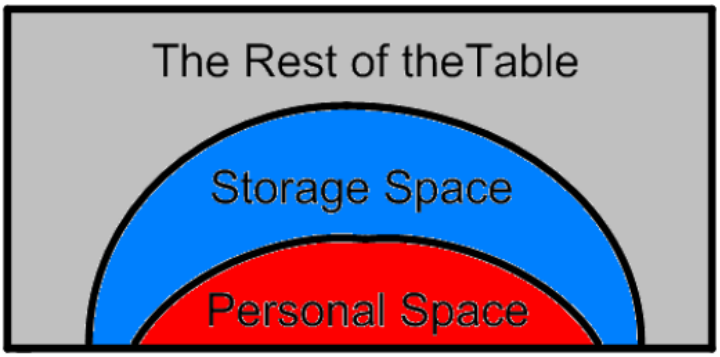

Figure 2.1: Reaching areas on tabletops.

2.1 Large direct-manipulation interfaces

The size of the screen plays a big role

It may sound obvious, but the size of the display in direct-manipulation interfaces plays a big role. When standing at a fixed position in front of a tabletop, humans have a certain range that can be reached, and everything that is beyond that radius can not be touched without changing the standpoint or without doing some acrobatic wrenches on top of the surface.

The Personal Space and the Storage Space

Toney and Thomas [2006] did some empirical testing to find out the dimensions of this direct reach envelope (also called Kinetosphere). They figure out two different areas. The first one, that they call *Personal Space*, is the area where users prefer working in. All the manipulations are preferred to be done here, because it is the most comfortable zone to interact in. Another space, called *Storage Space*, is an area where users tend to put their objects that they currently do not use, but that can be dragged into the Personal Space quickly. Figure 2.1 shows a sketch of these two spaces that were defined by Toney and Thomas [2006].

In order to calculate the speed of an interface, we can recur on Fitts's law, which has been proposed by Paul Fitts in 1954. The well-known formula looks like this:

$$T_{pos} = I_M * \frac{2D}{W}$$

I_M represents the index of movement, D stands for the distance to the middle of the target and W stands for the width of the target. Bi et al. [2013] propose a modified formula, which takes into account the standard deviation of the touch points and the absolute precision of the input finger. Nevertheless, their formula is still dependent on D, which means that if D increases, T_{pos} increases too. Therefore, it is advisable to place objects that will often be used in an area where they can be accessed quickly. When combining this with the spaces defined by Toney and Thomas [2006], it is recommended to put any kind of menus not too far away from the Personal space, where most action happens.

Place objects that will often be used in close areas

Other potential issues that may arise from using larger interactive table surfaces, like for example the display resolution, the work startegy or simply the visibility, have been compiled by Ryall et al. [2004].

2.2 Multi-touch gestures

On standard desktop computers, you usually have one mouse that controls one virtual pointer on your screen. Baecker [1995] defines it as a transducer from the physical properties of the world into logical parameters of an application. By its possibility to move in two directions, the mouse provides two spacial degrees of freedom.

The mouse has two degrees of freedom

Modern trackpads, which allow the simultaneous recognition of multiple fingers, can be interpreted as an increase in terms of degrees of freedom. However, as they are still dependent on the current position of the mouse pointer, they are still restricted. Therefore, a multi-touch trackpad can, in my opinion, more likely be considered as a surface that allows gestural shortcuts. The biggest difference to "direct multi-touch" is that trackpads can only change one single object at a time. So either this object needs to be selected first, in order for the system to know what it should manipulate, or it has to be defined in advance, so that all gestures will be applied to a specific object. Editing multiple objects simultaneously is not possible.

A multi-touch trackpad allows gestureal shortcuts

Real multi-touch became popular with the appearance of the iPhone

On the other hand, real multi-touch devices which translate the exact position of each single finger to the running application, allow simultaneous manipulation. These kind of devices have actually already been investigated by Mehta et al. [1982], but only became really popular with the appearance of the iPhone in 2007. In these cases, a touch is mapped to the object that lies directly underneath the source of input, and so is the gesture that is being recognized.

Tabletops encourage the collaborative manipulation

Theoretically, a multi-touch system can differentiate an infinite number of simultaneous touches on the surface, which implies its usage as a collaborative device and its denomination as a multi-user system. This applies mostly for large multi-touch surfaces and not for smartphones or tablets that are intended to be operated privately. In the case of tabletops, the additional people around the device represent a new problem which is called the touch-to-person-mapping. In order to recognize the right gestures, the system ideally knows which touch belongs to which user. Here, we still have to differentiate between systems that *distinguish* users (Dietz and Leigh [2001], Zhang et al. [2012], Annett et al. [2011]) and systems that *identify* users (Meyer and Schmidt [2010], Roth et al. [2010], Holz and Baudisch [2013]).

There is a lot of leeway for new gestures

End users should define their own gestures

Once the touches that belong together have been identified, their meanings need to be interpreted. On the one hand, there are some gestures like the two-touch pinch-to-zoom gesture, that have already become widely accepted. On the other hand, there is still a lot of leeway for new ones that may only be used in specific use-cases. Kammer et al. [2010] try to formalize gestural interaction on multi-touch surfaces, while Fuentes et al. [2011] enlarge the set of gestures for general actions such as selecting, moving, rotating and resizing by a set for more specific actions like editing or deleting objects, modifying object properties or undoing and redoing. Nevertheless, everybody thinks differently, everybody has had different experiences with multi-touch surfaces and not everybody is free of cognitive or motion impairments and therefore Jégo et al. [2013] propose to let the users define their own gestures.

The two papers in the last section by Kammer et al. [2010] and Fuentes et al. [2011] investigate gestures that rely on only one hand, however tabletops are big enough and do not need to be carried like tablets, so the second hand can also be used simultaneously. Two-handed input techniques have the advantage that they are in line with the bimanual skills used in the physical world, like Jiao et al. [2010] mention it in their paper. And even though Buxton and Myers [1986] proved that bimanual manipulations in Human-Computer-Interaction (HCI) can result in a higher and a more efficient performance, Kabbash et al. [1994] mentioned a few years later that if the two-handed input technique is not designed properly, there will not be any gain in bimanual gestures. A first investigation of bimanual gestures comes to the point that they need to be divided into *asymmetric* and *symmetric* gestures.

If bimanual gestures are not designed properly, there is no gain

Asymmetric gestures, which could be compared to peeling a banana, are actions where the left hand performs a different task than the right hand does. These techniques are usually guided by the Kinematic Chain (KC) model invented by Guiard [1987]. This theory is based on the fact that humans have a dominant hand and a non-dominant hand. The non-dominant hand, usually responsible for the coarser movements, sets the frame-of-reference as Hinckley et al. [1997] called it (holding the banana), in which the dominant hand, responsible for the more delicate tasks, works (the actual peeling of the banana). In the case of tabletops, the non-dominant hand could select a certain object on the screen, whereby the dominant hand manipulates it.

Asymmetric gestures

Symmetric gestures, like rolling out dough, have the main advantage that the user does two similar things with both hands, so that the cognitive load is reduced compared to asymmetric gestures. However, Balakrishnan and Hinckley [2000] mention that for symmetric tasks, it is important to distinguish between *task assignment* and *task performance*. What the author wants to say by this is that the tasks both hands should perform can be identical. Although this does not imply that they are performed together, because they may be executed sequentially.

Symmetric gestures

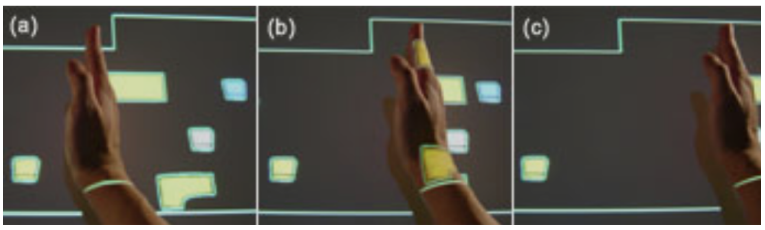

Figure 2.2: Wu and Balakrishnan [2003] show how multiple virtual objects can be pushed aside at once using the upright hand.

Touches can also be performed with whole hands

Usually, when thinking about multi-touch surfaces, one associates each touch to a finger. Nonetheless, touches can also be performed with the whole hand. For example, when you have a few balls lying around on a table, the natural gesture to congregate all the objects would be to position your hand upright on the table, thumb on top, maybe curve it a little in a "C-shape" and drag it over the table. So why not doing the same movement with a large number of objects on a virtual screen? Figure 2.2 shows an example of how the sequence could look like when pushing multiple virtual objects to the side.

Objects in real-life depend on physical constraints

Besides these gestural interactions that are based on the whole hand, one major difference between the moving of virtual objects and objects in real-life is that the latter ones do not glide fluidly over a screen, but depend on physical constraints, like the friction between them and the underlying surface or the force of the spring that has been built into a button. To simulate these physical properties, Cao et al. [2008] introduce ShapeTouch, which infers virtual contact forces from contact regions and motion to enable interaction with virtual experiences of interacting with real physical objects. Figure 2.3 shows an example on how flicking works in ShapeTouch depending on the amount of fingers used. When only flicking with one finger, like in (a), the card is only sliding slowly a few centimetres into the direction the user is pushing, but when simulating a bigger force, like in (b) and represented by the use of more fingers, the card will slide quicker to the front and also further away.

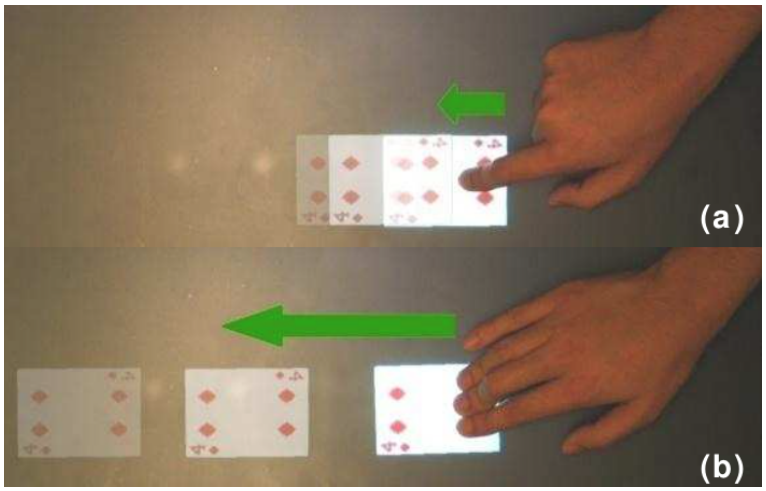

Figure 2.3: More fingers represent a bigger force. (a) Small contact flicks an object slowly. (b) Large contact flicks an object quickly.

Editing 3D objects on 2D screens

Even though my focus lies on 2D applications, helpful gestures for my later implementation could also be found in papers that handle the manipulation of 3D objects. Sun et al. [2013] for example present a multi-touch interface for fast architectural sketching and massing. As there is a third dimension, new gestures had to be considered. Figure 2.12 shows a few examples of new gestures that allow a faster prototyping of modern building structures. Other example papers exploring multi-touch gestures in the third dimension would be *Paper3D* by Paczkowski et al. [2014], *The design and evaluation of 3d positioning techniques for multi-touch displays* by Martinet et al. [2010] or *A multi-touch 3D set modeler for drama production* by Cardinaels et al. [2008].

2.2.1 Limitations

Users do not all have the same mental model

When talking about all the gestural possibilities, one also needs to mention some of their limitations. First of all, not all the people have the same mental model, which means that when you ask them to perform a suitable gesture for a certain situation, not all would do the same one, unless it is a gesture that has already gained acceptance, like the well-

known two-finger pinch gesture. This implies that the user will need to memorize all the new and application-specific ones, which will result in a decrease of performance as Jégo et al. [2013] observed in their study. If it is similar to the movement the user would do in real life to manipulate the object that is presented on the screen, she will get used to it with a small amount of practice. However, if the interaction is so unique, it will be hard for the user to perform it later on, on the fly, without recalling it explicitely. Especially when performing a similar gesture they performed on another device, people tend to reuse their acquired experience and translate it to the new device, which Jégo et al. [2013] call the *propagation effect*.

Also in real-life people need some practice

A second limitation that has already been broached before, is the symmetric, respectively asymmetric interaction using both hands. Hinckley et al. [1997] mention that a bimanual manipulation usually consists of a non-dominant hand that sets the frame-of-reference and the dominant hand that performs in it. So when thinking about new gestures, one can not demand from the end user that she performs two different fine motor skilled tasks with both hands simultaneously, which both need her attention.

The non-dominant hand sets the frame-of-reference for the dominant hand

Another limitation is not the motoric constraints between both hands, but the ones between the fingers. As Olafsdottir et al. [2014] point out, motor human factors limit the number and the complexity of shapes users are able to execute and therefore, anatomical properties and constraints have to be taken into account. Muscles shared between fingers, and overlapping cortical representations lead to the inadvertently move of one finger when manoeuvring an adjacent one. Häger-Ross and Schieber [2000], Yu et al. [2010] and Zatsiorsky et al. [2000] all come to the same conclusion, that the motion of fingers are bound to each other and that none of them can move independently, so that certain amount of gestures are simply not possible because of anatomical reasons.

Motoric constraints between fingers

2.3 Menus on multi-touch surfaces

No matter if we are using a Windows-based, a Mac-based or Linux-based operating system on our desktop computer, we always expect a hierarchical linear menu on the top edge. Additional menus, mostly representing shortcuts of things that can also be found in the main menu on top, are usually placed somewhere around the main workspace. Context-menu are usually available when right-clicking on the object of interest.

Users expect a menu on the top edge

For mobile devices we encounter a different setup, which depends on the screen size. Especially smartphones can no longer show menus with fingertip-size targets as they would already cover a considerable part of the screen. An additional problem is that even though smartphones and tablets allow multi-touch interaction, most of the time the second hand is only used for support. Therefore researchers thought about new techniques how to operate on such devices and how to display menus on them.

Different menus on smartphones and tablets

2.3.1 Menu types

Before talking about the specific menus used on the different multi-touch devices, I first have to give a small overview of menu types that exist.

Drop-down menu (”Popdown menu”)

Drop-down menus are menus displayed on demand on mouse click or hover. In the case of Microsoft computers for example, they are bound to a menu bar, a graphical control element placed on top of a window just underneath the title bar and whose content is application-dependant. They are normally hidden from view and therefore are an efficient means of conserving screen space (Microsoft [2014]). Drop-down menus can also have submenus.

Context menu ("Popup menu")

Context menus depend on the current context

A context menu is the global term for a menu in a graphical user interface, that appears upon user interaction; often right-click on desktop computers. It is called context menu because the choices that are displayed in the menu depend on the current context. These kind of menus can either be linear or circular. In linear context menus, all items are listed from top to bottom, sometimes enumerated with an index number. Circular context menus, also called *pie menus*, are menus that consist of several slices, usually four or eight, all equidistant to the center.

A first comparison of linear vs pie menus has been done in 1988 by Callahan et al. [1988] and shows an increase in performance of 15% less time and a reduction of selection errors for pie menus.

Marking menu

Experts just need to draw small marks

Marking menus are special cases of pie menus. By performing a touch gesture for more than a certain amount of time, (Kurtenbach and Buxton [1994] suggest a third of a second), a pie-shaped menu will appear around the location of the touch. Now the user can move her touch to the area of interest (see Figure 2.4 (a)). Experts, that know where the different menu points will be located around the touch, can also use a shortcut by simply drawing a mark. A mark is drawn by pressing the pen down and immediately moving in the direction of the desired menu item (see Figure 2.4(b)).

Marking menus are helpful for beginners and experts

The advantages of these kind of menus are, that they can be shown at any location defined by the end user. Combined with their reduced size, it is very improbable that something of interest will be occluded, like it could be with any drop-down menu. Moreover, marking menus offer visual feedback, which is essential for novices, but also offer the possibility to draw marks, which results in an increase of speed for experts who can recall the location of the item in the menu. This increase of speed can easily be demonstrated using Fitt's Law. In marking menus, all items have

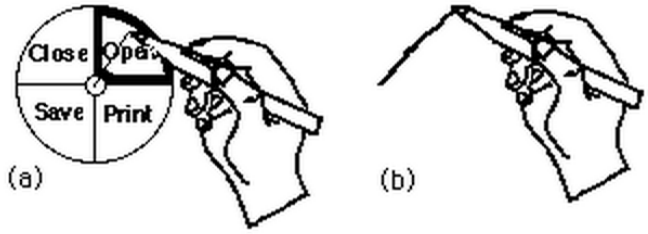

Figure 2.4: Marking menus: Kurtenbach and Buxton [1994] show the selection using a radial menu (a) and selection by drawing a mark (b).

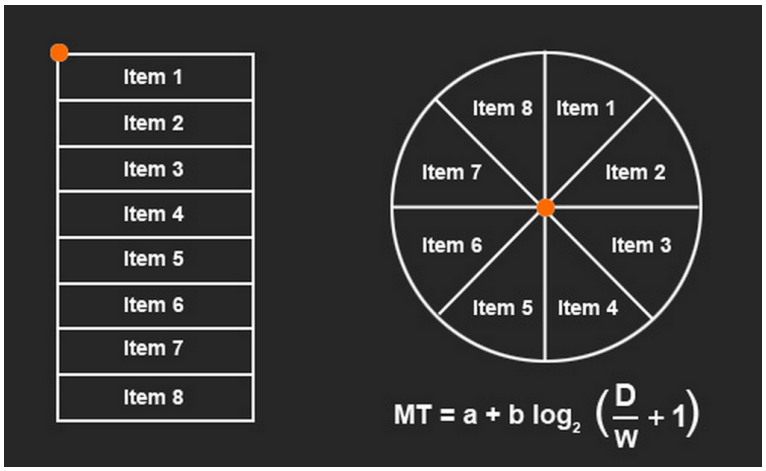

Figure 2.5: A visual representation of the difference between the Movement Time in linear menus and marking menus (Felix [2009]).

the same target distance (D) and the same target width (W), so that the average Movement Time (MT) will remain constant, no matter how many slices exist (see Figure 2.5).

There are only two main disadvantages for marking menus. The first one is that you can not display them close to a border, because then part of them will be occluded. The second one is that the amount of items is limited. Theoretically, an infinite number of items is allowed, however the more items you have, the harder it will get to hit the right one.

Marking menus also have disadvantages

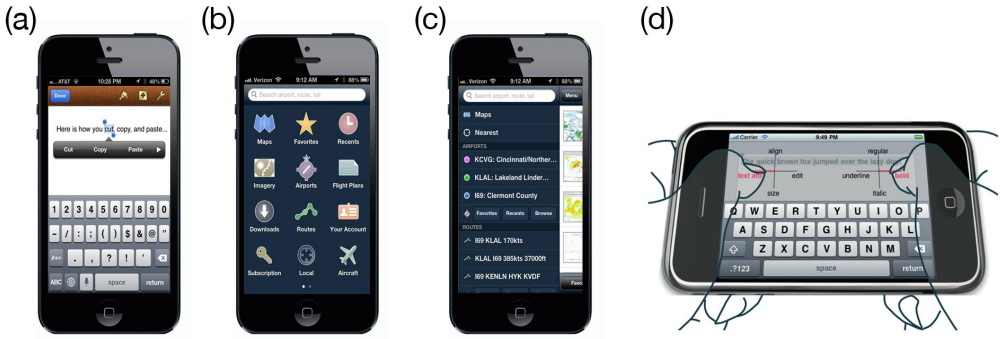

Figure 2.6: Menus for smartphones: a) the standard popover menu, b) a menu that is shown on a new view, c) a menu that is actually placed behind the main view, d) a bimanual marking menu.

2.3.2 Menus on smartphones

A new view for menus on smartphones

For smartphones, there exist many different menu types. First of all, because of its reduced screen size, many developers choose to put a menu into a new view if it is too big (Figure 2.6 (b)). Menu bars are even often placed behind the main view (Figure 2.6 (c)), which slides over when the user wishes to see the menu. If the menu only consists of a few choices, letting a popover appear is also often considered as an option (Figure 2.6 (a)).

Marking menus on smartphones

Kin et al. [2011a] even implement a possibility to use a bimanual marking menu on a smartphone (Figure 2.6 (d)). However, their user study showed that even though the actual movement time was faster than normal marking menus, the total time was approximately the same. This is due to the slower reaction time, indicating that users spend more time remembering and planning their strokes when coordinating simultaneous motions of two hands.

2.3.3 Menus on tablets

As tablets are similar to smartphones, their menus do not differ much. The most effort to explore new menus has been done to make use of the multi-touch possibilities

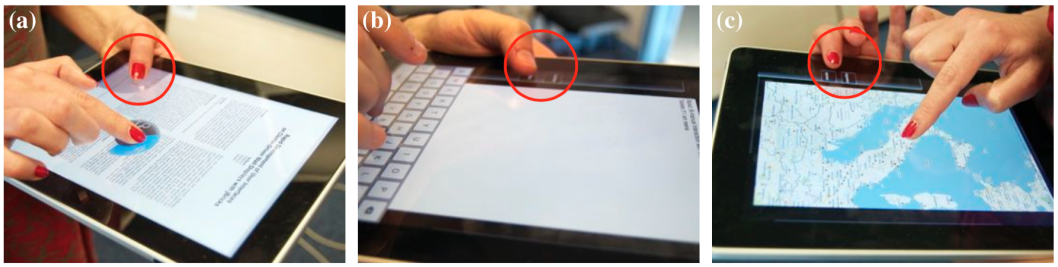

Figure 2.7: Bimanual interaction with BiPad: a) navigating a PDF, b) shifting to uppercase, c) zooming on a map. The non-dominant support hand can tap, make gestures or perform chords, thus modifying interaction by the dominant hand.

tablets offer. Just like Kin et al. [2011a], Foucault et al. [2014] and Wagner et al. [2012] make use of the second hand, *the supporting hand*, to switch between different menus. Figure 2.8 shows an example of SPad by Foucault et al. [2014], while 2.7 shows the possibilities offered by BiPad (Wagner et al. [2012]).

Use the supporting hand for menus

Another new technique is FastTap by Gutwin et al. [2014]. FastTap uses thumb-and-finger touches to show and choose from a spatially-stable grid-based overlay interface. This overlay should allow the user to perform shortcuts, just the way she would do it using a physical keyboard. Similar to marking menus, novices still have a visual feedback of all the possibilities, whereas experts only need a single quick thumb-and-finger tap.

FastTap uses thumb-and-finger touches

2.3.4 Menus on tabletops

Menus on tabletops do not need to be placed at the lateral edge of the device. As these bigger screens do not need to be carried, one can not assume that the user's hand will be located in that area. To the contrary, when operating on a multi-touch table, both of the user's hands will change their positions constantly. Therefore, classical menu bars are not suitable for tabletops. On desktop computers, the speed- and acceleration-settings of the mouse make it possible that by a small hand movement, the top of a window or the top of the screen can already be reached. However, for tabletops the actual distance always needs to be covered.

Classical menu bars are not suitable for tabletops

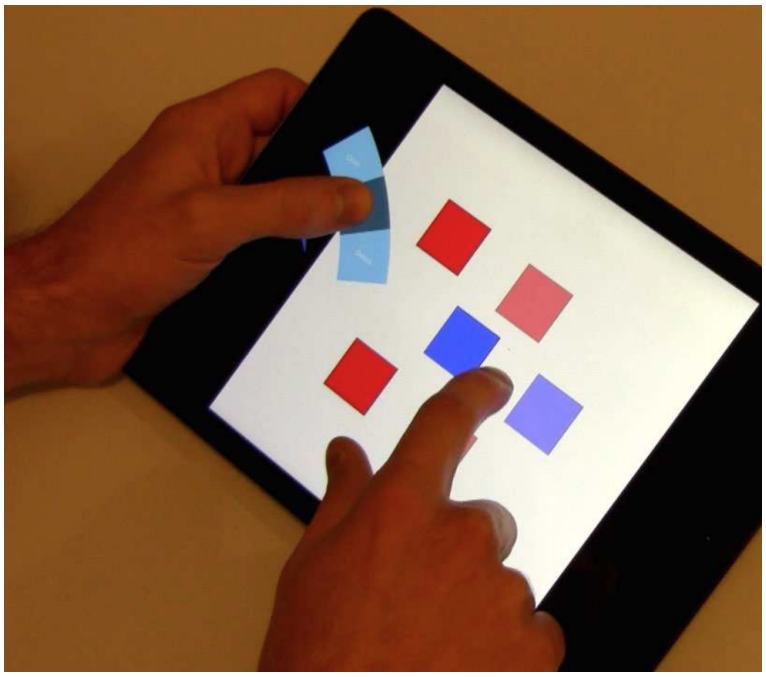

Figure 2.8: Copying multiple objects with SPad.

Finger-count shortcuts

An alternative when you still want to use menu bars is introduced by Bailly et al. [2010]. The first technique they present is called *finger-count shortcuts*. The system will count the number of fingers that touch the surface for each hand. Like this, 5x5=25 different menu items can be specified with both hands. In general, the non-dominant hand defines the menu of the menu bar, whereas the amount of touches of the dominant hand is associated with an item in the currently selected window.

Radial stroke shortcuts

The second technique presented by Bailly et al. [2010] is called *radial stroke shortcuts*. This time, instead of placing a certain amount of fingers on the interactive surface, the user draws marks on the screen, just like in marking menus. The orientation of these marks define the selected menu point. Again, the non-dominant hand selects the menu, whereas the dominant hand selects the item in the menu. Clearly, both of these approaches interfere with other gestures performed on the background, and therefore represent a certain limitation.

Context menus are more suitable. On direct input surfaces, occlusions created by the user's hand decrease in-

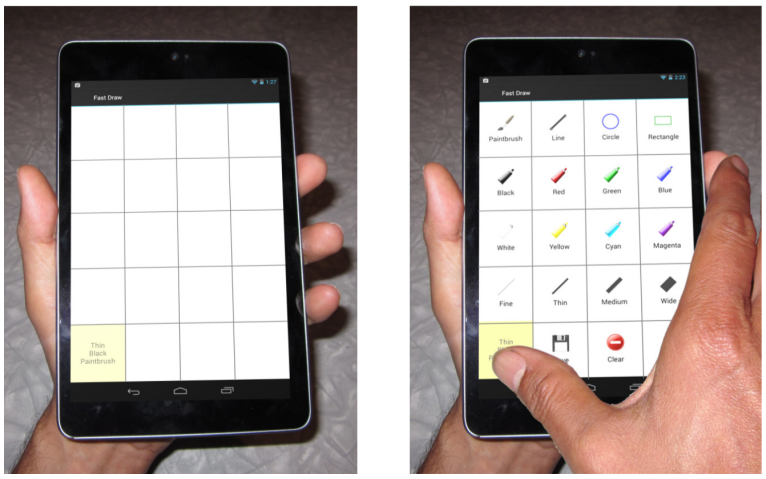

Figure 2.9: FastTap: The default state and the grid overlay after touching the activation button.

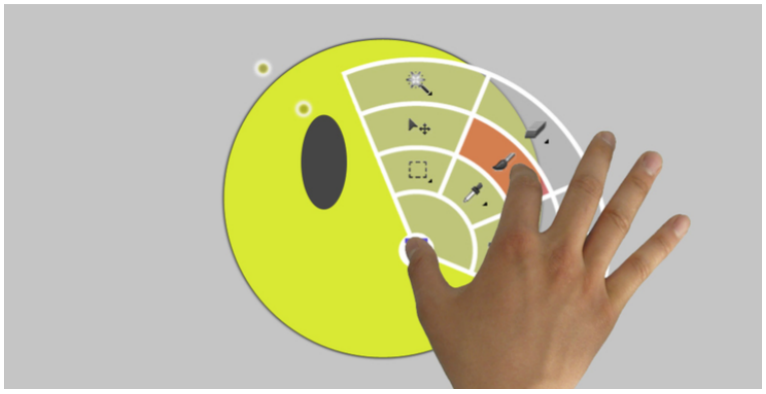

Figure 2.10: The user selects the paint brush by anchoring her thumb anywhere on the screen and by touching the brush area with her index.

teraction performance with menus, therefore, Brandl et al. [2009] present an adapting menu that avoids occlusion. In the study they present, the user uses a pen to interact with the screen. Due to fatigue effects, people tend to rest their hand on the surface, so that the system can react to the pen and hand position by presenting the pie-menu accordingly.

Context menus are more suitable

Banovic et al. [2011] go one step further, and investigate how to design context menus for efficient *unimanual* multi-

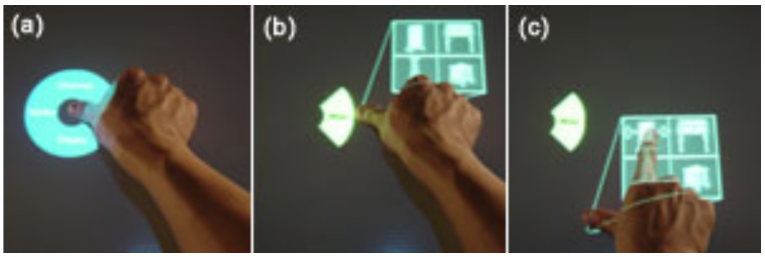

Figure 2.11: FurniturePalette tool. (a) A double tap on the table brings up a context-sensitive menu. (b) Sliding the finger in one of the four directions causes a corresponding toolglass to be attached to the finger. (c) A second finger is used to make a selection within the toolglass.

The efficient design of context menus

touch use. Tabletops encourage you to do multiple things at the same time, what leads to the fact that the user can not always use his second hand as an aid to do selections in a menu. When selecting targets with only one hand on a pie-menu, one can either use single-finger input or multi-finger input as it is shown on Figure 2.10. Their study shows that when using multiple fingers simultaneously, the performance of target selection can be improved, but the error rate increases.

Choose categories with the thumb and then items with the index

A mixture between the unimanual multi-finger input method, presented in the previous paragraph, and a standard marking menu is proposed by Wu and Balakrishnan [2003]. A pie-menu is shown when double-tapping anywhere on the screen with your thumb and leaving it on the screen (Figure 2.11 (a)). By sliding the thumb to one of the four cardinal directions, a new semi-transparent toolglass (Bier et al. [1993]) appears, which is bounded to the thumb's position (Figure 2.11 (b)). This toolglass contains several items, that can now be selected with any other finger of the hand (Figure 2.11 (c)).

2.4 Object manipulating applications

There exist a bunch of similar applications that manipulate virtual objects on tabletops.

One, which is called *Eden*, focuses on the set construction of virtual scenes. Kin et al. [2011b] define a set construction as the process of selecting and positioning virtual geometric objects to create a virtual environment used in a computer-animated film. The use of multi-touch should ease the process of constructing these virtual sets.

Eden lets you create virtual scenes

At the end of their paper, Kin et al. [2011b] note a few lessons they learned during the construction of their multi-touch application. The most important ones for my later work are the following ones:

Reduce simultaneous interactions
The human being might have two hands, but if both do separate interactions, she can usually only focus one one, especially if there is no haptic feedback. Therefore, using two different hands to manipulate two different objets should be limited.

Do not overload one hand
Because of the anatomical properties of the human body, some gestures that use multiple fingers of the same hand are sometimes hard to perform. Instead, splitting up a gesture to both hands can result in an increase of speed.

Consider occlusion
In contrast to the small mouse pointer, a hand can occlude important parts of the screen. For that reason, static touches should be releasable once a gesture is recognized, as Wu et al. [2006] mention.

Another application called *RoomPlanner* focuses on creating virtual scenes too. The paper was already composed in 2003, three years before Jeff Han's famous TED talk about multi-touch sensing. Wu and Balakrishnan [2003] lay their focus on new interaction techniques for multi-touch tables. Their example application mainly differs from Eden in a way that first, it is only two-dimensional so that they do not have to think about gestures that manipulate objects in a third dimension. Secondly, it is built to be used by multiple users simultaneously, so that besides the interaction techniques, they also explore visualization techniques for supporting shared spaces, awareness and privacy.

Roomplanner focuses on creating virtual scenes

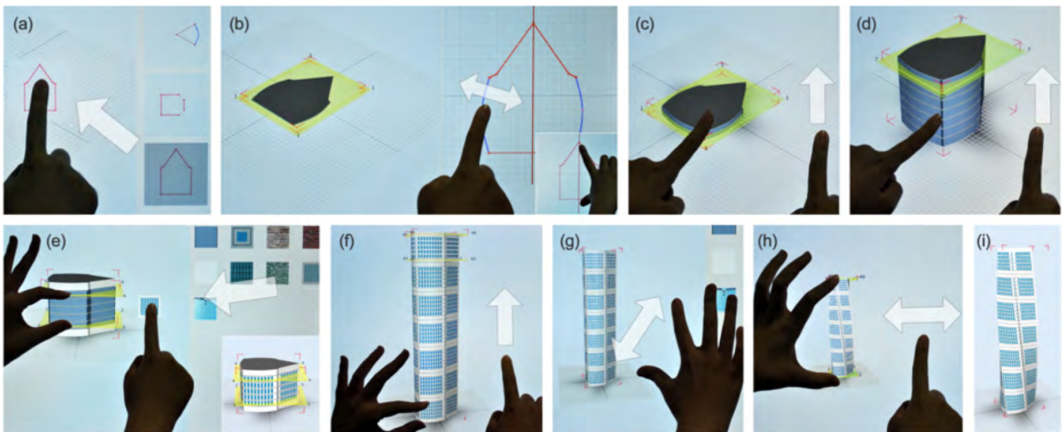

Figure 2.12: (a),(b),(c) and (d) show how to create the base floor plan, and how to extrude it. In (e), a texture is added to the building, in (f) the lower floors are being copied on top of them, and finally (h) shows how to add a torsion to the building. The classical two-finger pinch gesture is also applicable in order to zoom in or zoom out (g).

The most important insights Wu and Balakrishnan [2003] could find were:

Consider occlusion
When people were operating the pie-menus, the tapping hand sometimes occluded some of the displayed menu items. A solution could be to use only a semi-pie menu, instead of a complete circle. By recognizing which hand performs the action, one could display the items to the right or the left of the hand, which could facilitate other functionalities.

You can not envisage the used finger
In the RoomPlanner application, they implemented a pie-menu that appears when tapping twice with the thumb on the screen. Depending on the menu point you drag your thumb to, a new widget menu appears that can be manipulated with your index finger (Figure 2.11). As soon as the thumb is released, the menu should disappear. However, if you invoke the pie-menu with your index finger, you will run into trouble when trying to select the widgets.

A third application focuses again more on the manipulation of three-dimensional objects, sky-scrapers to be more

precise. It lets you sketch the 2D contour of a base floor plan and extrude it to model a building with multi-floor structures. After the initial design, the user can edit the structure of the building by applying various multi-touch gestures. The overall appearance can also be enhanced by dragging and dropping image textures from the library onto the building.

A software to create sky-scrapers on tabletops

A last paper that needs to be mentioned is called *Collaborative Concept Mapping at the Tabletop* and has been published by Martínez Maldonado et al. [2010]. Unfortunately, it focuses more on the educational area, so that the multi-touch gestures they use are not explained in detail. However, it shows a way how conceptual maps can be presented on a multi-touch table, how menus could be presented, and how the deletion of objects could be performed.

How to present conceptual maps on multi-touch tables

Chapter 3

Graphs

Graphs, often also called diagrams or charts, are probably the most popular and convenient medium to visualize information in a quick and clear manner. However, this information can communicate different facts, and depending on its characteristics, we use different kind of graphs. For example, if a physicist wants to show a certain value that changes over time, she will probably use a line chart. Or when a newspaper wants to visualize the allocation of seats in a parliament, it will probably use either a pie chart or a donut chart. Other common graph types are bar charts, flow charts, trees, venn diagrams or network diagrams. But actually, there exist plenty of other types, whereby everyone of them has its own history.

Each graph has its own purpose

Nicole Oresme (1320-1382), a Frenchman who was a counsellor at the court of Charles V, is considered to be the first person to use a graphical representation of a functional relationship between two variables. In his publication "de Latitudinibus", which can be translated as "The Latitude of Forms", he plots velocity of a constantly accelerating object against time (Figure 3.1).

Nicole Oresme was the first one to use graphs

As Oresme chose the bar chart to represent one variable over time, and did not pick it to show comparisons among different categories, which is considered today the definition of a bar chart, it is William Playfair (1759-1823) who is credited with the invention of it. By illustrating the imports

William Playfair is credited with the invention of the graph

Figure 3.1: Nicole Oresme already used bar charts in his publication "de Latitudinibus", dating from the 14th century.

and exports of Scotland in his book "The Commercial and Political Atlas", using elongated rectangles, he is the first person to split numerical data into discrete groups and plot them afterwards. Figure 3.2 illustrates one of the bar charts he used.

Today, in the 21st century, bar charts are still very popular, but a wide variety of new graphic visual representations have been derived to meet the different requirements

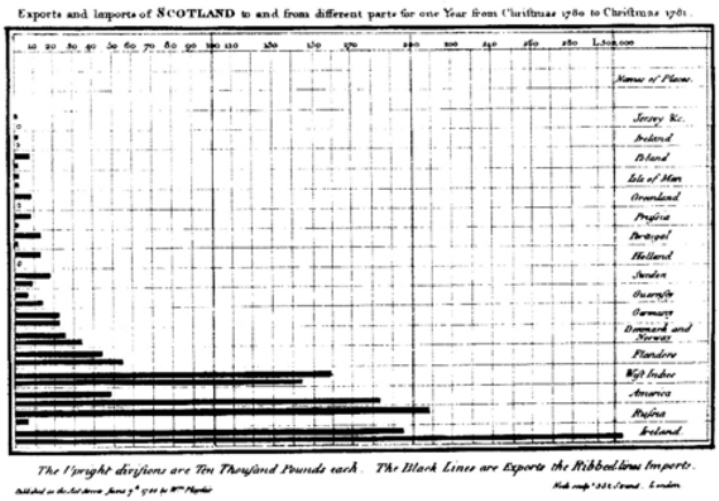

Figure 3.2: A bar chart by William Playfair, representing the imports and exports of Scotland (reprinted from JPowered).

in order to represent data and information in an adequate manner. The chosen graphs are very dependent on the field they are used in. Pie charts for example, which by the way have also been invented by William Playfair, are most of the time used to visualize distributions. Another example would be flow charts, which are very popular in computer science, showing the workflow of algorithms or other kind of processes.

Graphs depend on the field they are used in

However, charts are not only used to present complex data quickly and clearly, but are also helpful in solving of mathematical problems. In *Graph theory*, the graphs model the pairwise relations between objects. These objects consist of *vertices* or *nodes* whereas the relations between them are represented by *edges*, which can be directed (arrows) or undirected (lines). In his book *Introduction to Graph Theory*, Trudeau [2013] gave the following mathematical definition of graphs: "A graph is an object consisting of two sets, called its vertex set and its edge set". It will be these kinds of graphs that represent my area of interest, graphs which have a certain amount of nodes, and which are connected by edges, modeling the relations between these nodes.

Graphs also help solving mathematical problems

It is often hard to tell if the user planned to do her graph in a certain way, or if she was restricted by the software

As I would like to manipulate not only the position of the vertices and edges, but also their appearance on a multi-touch surface, I first need to examine their characteristics, and identify which ones are actually manipulated by the users. Even though it is hard to determine if the creator of a graph intended to visualize a property in a certain way, or if the software at her disposition restricted the final outcome, I adopted the following approach to find out what is absolutely necessary for individuals when they plan to create graphical illustrations.

3.1 Approach

I skimmed the graphs of all the papers of two conferences

After skimming all the proceedings of the *Conference on Human Factors in Computing Systems (CHI)*, that have been accepted in the year 2014, and after extracting all the graphs that corresponded to Trudeau's definition, I analyzed their common points. As already mentioned before, it was often hard to identify if the creator of the graph was restricted by the software, or if she could really map her idea onto the screen the way she wanted to. Another detail that was sometimes hard to identify was if some objects of the graph were hooked together or if they were just "flying around" and placed close together so you could think they belong together.

In order to confirm my outcome, I did the same thing with another set of proceedings from another conference where I was hoping to find a lot of graphs. Therefore, I chose the proceedings of the *Workshop on GRAph Data management Experiences and Systems*, which was held in June 2014. The results of both studies can be found in the next section.

	Amount	Percentage
TOTAL	76	100%
Shapes		
Rectangles	54	71%
Ellipse	26	34%
Images	23	30%
Database	4	5%
Other	4	5%
Color		
Fill Color	40	53%
Stroke Color	16	21%
Stroke style		
Dashed/Dotted	6	8%

Table 3.1: Some details about the graph nodes used in the papers of the Conference on Human Factors in Computing Systems '14

3.2 Results

3.2.1 Conference on Human Factors in Computing Systems '14

In my first set of proceedings, which consisted of 465 papers, I was able to identify 76 graphs that matched my requirements. Noticeable was that most nodes were either rectangles, ellipses or images. Other kind of shapes, like hexagons, the database symbol, or any other specific flowchart symbol were very rare. About the lines, one can say that directed graphs are way more popular than undirected ones, but still every fourth graph used arrow-less lines. A complete list of the interesting characteristics can be found in Table 3.1 and Table 3.2.

Most people only use rectangles and ellipses

	Amount	Percentage
TOTAL	76	100%
Type		
Directed lines	68	89%
Undirected lines	18	24%
Stroke		
Stroke Color	32	42%
Stroke Width	12	16%
Stroke style		
Dashed/Dotted	16	21%

Table 3.2: Some details about the graph lines used in the papers of the Conference on Human Factors in Computing Systems '14

3.2.2 Workshop on GRAph Data management Experiences and Systems '14

In my second set of proceedings, which consisted out of 12 papers, I could identify 19 graphs that matched my requirements. This time, ellipses were twice as popular as in the CHI papers. Rectangles were not as dominant, but still used in more than half of the papers. And bizarrely, all the nodes were shapes, not a single image was used as a node. On the first glance, one realizes that the graphs were kept much more simple than in the previous set, without many stroke modifications, and especially many images (Two graphs contained images, but they were not used as nodes). A complete list of the interesting characteristics can be found in Table 3.3 and Table 3.4.

In this conference the graphs were kept simple

3.3 Conclusion

First of all, I have to say that is is not a complete and reliable study. This review of graphs helped me to discover what people actually tend to use when creating charts so I could offer them these features in my subsequent application. However, when you want a more precise study, you should observe users in the field and not only rely on this

A better study can be conducted in the field itself

	Amount	Percentage
TOTAL	19	100%
Shapes		
Rectangles	10	53%
Ellipse	12	63%
Images	0	0%
Database	1	5%
Other	3	16%
Color		
Fill Color	14	74%
Stroke Color	0	0%
Stroke style		
Dashed/Dotted	1	5%

Table 3.3: Some details about the graph nodes used in the papers of the Workshop on GRAph Data management Experiences and Systems '14

outcome. Often, I did not know how to classify certain objets of the graph. "Is this now a node with text, where the stroke has been removed, or is is just a plain text that has been placed on the background?" was only one of the questions I had to ask myself a few times. So it is important to underline that this study is very subjective.

As I already mentioned, I tried to find the most important features in order to please the majority of my later users. Nevertheless, when you plan to write a complete program, where everybody should be satisfied, you can of course not leave away features that are only used by every tenth designer.

Furthermore, I did not take care of textual properties in this review. The characteristics of labels are very limited, as you can only change the font size and the font color. The interesting thing to know, would be if in the mental model of users, they attach texts to nodes or lines, or if they are stand-alone objects, whose position is relative to the background.

Do users bind labels to nodes or lines?

The few things that can be said for sure, and that I finally took from this review, were:

	Amount	Percentage
TOTAL	19	100%
Type		
Directed lines	15	79%
Undirected lines	4	21%
Stroke		
Stroke Color	1	5%
Stroke Width	0	0%
Stroke style		
Dashed/Dotted	2	11%

Table 3.4: Some details about the graph lines used in the papers of the Workshop on GRAph Data management Experiences and Systems '14

Recatngles and ellipses are dominant

People tend to use only the basic shapes in most of the cases, which are the rectangle and the ellipse. In my own work, they will be absolutely necessary, however other shapes like pentagons, hexagons, stars or freeform polygons do not represent a must. They are nice features, and the software could be extended by them, however users can also go without them.

People often use different tools for different purposes

It is often the case that people do not only rely on their graph tool alone. Sometimes, they prefer creating some parts in another tool that gives them more freedom, or that is better suited for a certain situation. Another possibility is to simply download graphics from the internet in the form of raster graphics and import them later on.

Most of the time, edges start at vertices

Moat of the time, edges start at vertices and point to other vertices. Only in a few cases, where the designer wants to show an entry or exit point, arrows do not have a starting state or end state, but for the rest one can say that they are always bound to two states.

Colors are an alternative medium to show relations

Colors are a very powerful medium to show that things belong together without connecting them with a line or without putting them in a container shape. Especially the fill color has often been changed to show some kind of relation.

As a system developer, it is always important to know *what* users intend to to with your system. What features to they need to be able to fully express themselves? Additionally, it may be even more important to know *how* they intend to use the system. If the designer and the end user have different mental models on how the application should be operated, the latter ones will be quickly annoyed by the system because it does not do what they were imagining it would. In the upcoming chapter, I will take a look at people's approaches of creating graphical images in order to better understand their workflow.

This study was conducted to see *what* users need

Chapter 4

Pilot Study

In the last chapter, I tried to figure out *what* people need when they plan to create a graph. This chapter will focus more on *how* they create the different parts, and in what order they do it. I was hoping to identify sequences that could be quickened using the power of direct-touch combined with specific multi-touch gestures. In order to perform this study, I invited eight volunteers to the lab of our chair so they could participate in a two-part-study, whose procedure I will explain in section 4.2.

Now I identify *how* people use their graph creating software

4.1 Participants

A total of eight participants participated, six male a two female, between the ages of 23 and 30, all from the field of computer science. One person was left-handed, the others were all right-handed. Three of them claimed having used Omnigraffle[1], the software that will be used in this experiment, a few times before on a Mac. It were also these three, that had a lot of experience with other graphical tools like Adobe's Illustrator for example. Only one participant played around with the mobile version of Omnigraffle before.

Only one person was left-handed

[1] https://www.omnigroup.com/omnigraffle

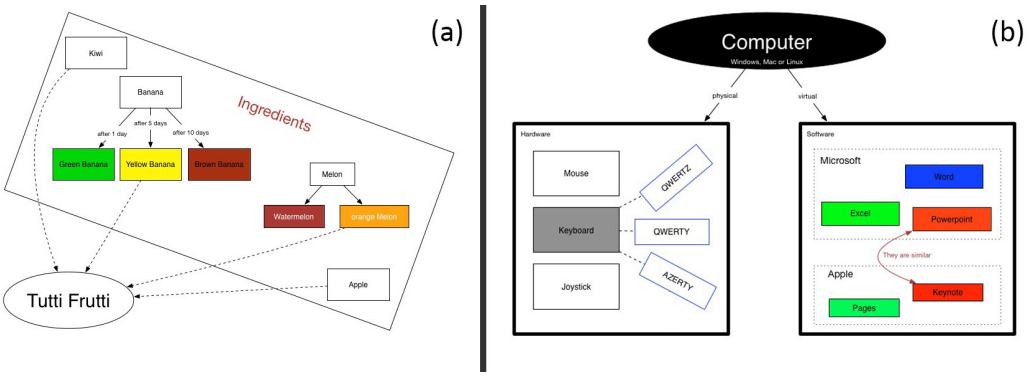

Figure 4.1: The two graphs the users had to recreate. (a) is more of a flow chart, while (b) is more of a graph that shows a tree structure.

4.2 Procedure

Users had to recreate two given graphs on the iMac and on the iPad using Omnigraffle

The pilot study consisted of two parts, whereby in both parts the user needed to recreate a given graph. In the first part, they had do recreate the graph on a 27" iMac using Omnigraffle, followed by the second part where they had to use the mobile version of Omnigraffle, running on an iPad of the fourth generation, to reproduce another given graph. Figure 4.2 shows the desktop and the mobile version of this software. Both of the graphs had almost the same amount of objects, whereas the first graph was more representative for a flow chart and the second graph showed more of a graph that included a tree structure (Figure 4.1). While one half of the participants started with the one graph, the other half started with the other one. There was also a very minimalistic graph with three nodes and two lines, which I used to do some preliminary explanations.

Definition: *TF-Graph and C-Graph*

TF-GRAPH AND C-GRAPH:
In order to not always need to write *Tutti Frutti Graph* or *Computer Graph*, I will use the abbreviations TF-Graph for the graph on Figure 4.1 (a) and C-Graph for the graph on Figure 4.1 (b).

The creation of both graphs should take around 15 minutes each, and the users were not given more than 20 minutes. They got this information in advance, and were also told

Figure 4.2: The desktop and mobile version of Omnigraffle.

that it would not be dramatic if they were not able to finish the graph in time. Most of the time, when a user was not able to finish her graph, it was due to the constant repositioning of the nodes. Between the operations on the iMac and on the iPad, there was always a small break, where the participants could relax, eat some sweets and have a drink. After each part, I asked them to note their concerns about the different applications and at the end each participant got rewarded with some chocolate.

Each task should not take more than 20 seconds

The experiment had the form of a silent observation, but the users could always ask me questions if they were stuck. Whenever I realized they had some troubles, I helped them by myself after a certain amount of time. Even though I took notes during the study, I also did two types of recordings to capture the users' performance. On the one hand, I used the videos to monitor the time they needed to complete a certain subtask, on the other hand, the recordings were useful to review some interactions. In both tasks, on the iMac and on the iPad, I did a screen recording, while during the iPad task I used an additional external camera that recorded the movements of the users' hands.

I did screen recording and gesture recording

4.3 Results

There are three different kinds of conclusions: Observations, Feedback and Strategies

For the results, I distinguish between 3 different kind of conclusions. First I focus on the objective observations that I did during the pilot study. Were there any special or conspicuous things that need to be retained? In the following section, I will amplify the subjective feedback the users gave me. And last but not least, I take a closer look at the construction strategies of the graphs by each user.

4.3.1 Observations

iMac

At least a few nodes were created first, before concentrating on the appearance

Group of nodes created together:
All of the users had the tendency to create at least a few nodes together before concentrating on their appearance. Some constructed all the objects at the beginning, dragged them to the desired spots and changed their properties there. Others only focused on a certain group of objects first that seemed to belong together. Also this time, editing the properties was only done when they had all the nodes they needed. More details about this can be found in section 4.3.3.

Beginners had troubles finding the right buttons in the menus

Too many menus are disturbing:
In Omnigraffle, you actually have a bar on the top, which lets you create new objects, either shapes, lines or labels. Additionally, you have another menu on the right side to edit the properties of the current object. These two different menus confused the novices at the beginning as they did not know directly where they could find the menu item they were looking for. One of them did not really get the concept at all, and had troubles throughout the whole study.

Double click to add text to a line:
When trying to add text to a line, every user, even the experienced ones, tried at least once to double click on the line. This however, created a new point that could be moved in

order to change the shape of the line. Text could only be added with the text tool.

Mode errors:
Whenever the user planned to draw a line, she pressed on the corresponding button, but, to be able to draw multiple lines after each other, the user needed to press this button twice, otherwise, she would draw one line, and the system automatically jumped back to the moving tool. So when trying to draw the next line, she started moving the object which should be the starting node of the line. This may be application-specific and will certainly be learned over time, but in my later work such mode errors should absolutely be avoided by providing a a more noticeable visual feedback.

The system always jumped back to the pointer mode, if not told explicitly

iPad

Mostly single-touch:
Even though the iPad supports multi-touch, almost all features were based on single-touch. The exclusive multi-touch gesture was the scaling and rotating of nodes. Besides this gesture, the only time the user really needed her second hand, was to type text or to zoom in or out. For the rest of the time, the participant either put her hand next to the iPad on the table, or even laid it into her lab.

Multi-touch was only used to scale and rotate nodes

Copying and pasting objects is time-consuming:
You always needed a long press on the object until a context menu appeared, where you had to select the copy function. Then you needed one more long press on the background, where you had to select the paste function in the context menu that appeared at that position. The default press duration of the long press is 0.5 seconds, so you need already at least one second to invoke the context menus. Added to this will then be the time that it needs to select the right item from the menu, which could be calculated with Fitts' law, depending on its position and width. All in all, copying items should not take too long, because it is a feature that is often used.

Copying and pasting was not as easy as on the desktop with a keyboard

Pop-up menus occluded parts of the working area

The keyboard and the pop-up menu from the menu bar are hiding important parts of the screen:
When the user invoked either the keyboard, by double clicking on an object, or when she opened the pop-up menu, by pressing on the item in the right upper corner, the background did not move accordingly. Parts that were in such areas of the screen where these pop-ups appeared, would be covered by them. This flaw can be perceived on Figure 4.2.

4.3.2 Feedback

iMac

The menu was too far away

Context menu & screen too big:
Two users complained about the constant changes between the objects, that were situated more or less at the center of the screen, and the menu that was on the right side of the screen. Both claimed that a context menu, which would appear where you invoke it, would have avoided all these long position changes.

iPad

There is no Ctrl- / Cmd-key that helps selecting multiple objects

Multiple objects can not be selected when they are included in another shape:
Multiple objects can be selected by touching the background, and then moving the finger diagonally over the objects you want to select. A selection area will appear, and the selected objects will be highlighted. However, when the objects are contained in another node, you need to start the selection on the background, outside this node, and you will then select this latter one unavoidably too as soon as the selection area intersects with it.

Small movements are difficult & Zooming is annoying:
All in all, what the users wanted to say here is that the reduced screen size of the iPad makes it hard to interact with it.

Other smaller concerns:
Many users complained about the unhandiness of the node rotations. Often the system did not recognize the double touch and the users became quickly frustrated. Furthermore, text labels inside a node could not be repositioned in this one. You were only able to change the text-alignment and font-size, but neither the position, nor the orientation.

4.3.3 Strategies

In this section, I will take a closer look at the strategies the users picked. Again, I will distinguish between the strategies on the desktop computer and on the mobile device, to see if there are some differences.

iMac

Kind after kind or subgraph after subgraph:
There are two main strategies on how the users iterated over their graphs, whereas the second strategy had many small variations.

Create all the nodes first

The first strategy, that has been done twice, was to create all the nodes at once, before focusing on their details or connections. While one user really only created the standard nodes first before concentrating on the details, the other one directly added the text to the nodes. In later iterations, they both then added the colors and the connections.

Create subgraph after subgraph

In the second strategy, the users tried to finish subgraphs first, before moving on to the next part. However, what defines a subgraph changed from user to user. Two of the three remaining users, that did the TF-Graph, finished each sort of fruit (colors, text and arrows) before moving to the next fruit. The last one created all the superclasses of the fruits first, before adding the subclasses to each of them. For the C-Graph, two of the three remaining people treated each of the two containers as a separate subgraph, and iterated over this one, while one thought that the three most

left shapes belonged together, and finished these first before addressing herself to the other three ones.

Top to bottom, left to right and back to front:
One can say that in general, each user worked through the graphs from top to bottom, left to right and back to front. Note that I only had users who also have a cultural background where they write from left to right. Each participant that had to recreate the C-Graph, started with the "Computer"-node before creating the two main container nodes. In the other graph, the lower left "Tutti Frutti"-node is kind of a stand-alone node and was recreated whenever the user was up for it. However, the other ones, no matter if they were created subgraph per subgraph or kind per kind, the working process was always from upper left to lower right.

Cultural background could play a big role

Concerning the z-order, only one participant used the "send to back"-command. All the other ones always created the objects from the back to the front.

Text before colors:
There was a slight tendency (6 out of 8) to add the text to nodes before the color. Especially in the TF-Graph, everyone first added the text.

Connections always at the end:
Connections were always created either at the end of each subgraph or at the very end of the graph. All the users who worked subgraph per subgraph, also added the arrows within this subgraph whenever they were done with it. Other arrows, that connected bigger parts, like the two arrows on top in the C-Graph, or the four arrows to the lower left node in the TF-Graph, were preferably done at the very end and always all together.

All the users did the connections at the end

iPad

Kind after kind or subgraph after subgraph:
I could again distinguish between the two main strategies, where the users either created all the shapes at once, or they

did them subgraph per subgraph. This time, three users created all the shapes first, while the other five laid their focus more on the consecutive creation of the subgraphs.

All shapes at once or subgraph after subgraph

Top to bottom, left to right and back to front:
For the C-Graph, three of the four participants created the top ellipse first, followed by the two big containers. One however, only did it at the very end. For the TF-Graph, the two first things every user did first, was to create the "Tutti-Frutti"-node and the big background node. After that, the objects in the upper left part, were always created before the objects in the lower right part.

Similar ordering strategy than on the iMac

Text before colors:
On the iPad, the participants even had more the tendency to create the respective label of each node before its color. Only one person, who iterated over the graph kind per kind, added the colors in the second iteration, before adding the text in the third one.

Connections at the end:
All the participants who worked subgraph after subgraph, also added their lines at the end of each part. Two of the remaining participants added the lines in their last iteration, while one participant directly added the lines in the second iteration when she had all the nodes without any colors or labels.

Connections after each subgraph

Similarities and differences

All in all, one can say that there are no big differences between the working strategies on the desktop and on the mobile device. This may correlate with the fact that the mobile versions of the same product are often only copies of the desktop versions. The interface is the same, the menus are very similar and the gestures are also not adapted to be real multi-touch gestures; most of the time, the mouse cursor is only replaced by a single finger touch. Altogether, we can recognize the following strategy patterns.

No big differences in the strategy when comparing the desktop computer to the mobile device

The recreation of the graphs was always a mixture between

left to right, top to bottom and back to front strategies. Sometimes, the one strategy was a little bit more dominant, while other times another one prevailed.

The main strategies can be put into two big classes: the one where the users created all the shapes first, before, in the upcoming iterations, concentrating on the colors, texts and arrows, or the other one, where the user always tried to finish each subgraphs first.

Visual correlations are more important than logical ones

The granularity of what the user perceives as belonging together, was diverse. However, what can be retained, is that visual correlations are more important than logical ones. Apart from the one user who create all the superclasses of the fruits first, all the other ones created the nodes that were close to each other one after the other. This is very conspicuous in the left container of the C-Graph. Even though the three right nodes are connected to the middle left one, and form a logical union, they were never created just after their parent, but always when the three left shapes were completed.

In my opinion people reference nodes rather with text than with colors. This is why most of them started adding the labels before the colors.

Arrows and lines are always added at the end; either at the end of the creation of each subgraph, or at the very end when all the nodes have been finished.

4.4 Conclusions

The strategies will help to compare them to the strategies on the tabletop

The strategy section did not help me that much to make the design decisions of my program. It will be more useful to compare the strategies between the desktop, the iPad and the tabletop to see if people tend to change their workflow on bigger multi-touch surfaces.

Context menus are necessary on large multi-touch tables

For the rest, the important realisations were that context menus can be helpful not only on big desktop screens , but also on large tabletop screens. Their closeness to the

user and their intended position represent a big advantage. Similar objects, or objects that belong together tend to be created together. Copying and pasting is also used a lot, and should therefore not consume too much time.

Chapter 5

Design and Implementation

Based on the results of my literature research, my graph review and my pilot study, I implemented a program, which allows the user to create graphs based on nodes and connections, and which is specially designed for big multi-touch tables. In my case, I deployed the application on Microsoft's Perceptive Pixel, a capacitive-based multi-touch screen, which has a display area of 72 x 41 inches and a 1920 x 1080 resolution. The device is arranged as a table, but could of course also be mounted to a wall. Each orientation has its advantages and disadvantages, like for example the annoying "gorilla arm" on vertical displays, which causes pain in the shoulder when the user can not rest her arm on the surface. The disadvantage of horizontal surfaces is that the user could develop a stiff neck from looking down all the time. Rogers and Lindley [2004], Morris et al. [2008] try to find out what is the best way to operate such big multi-touch displays whereas some even try to build on the advantages of each orientation, the vertical and horizontal (Weiss et al. [2010], Muller-Tomfelde et al. [2008], Wimmer et al. [2009]). Nevertheless, my software has been optimized to be used on a horizontal surface.

The multi-touch surface could either be mounted horizontally or vertically

Furthermore, when talking about multi-touch surfaces, one also thinks of collaborative interaction. Setups where the system has to differentiate between various users by de-

Multi-touch tables are often collaborative workplaces...

tecting which touch belongs to which user need even more complex algorithms than the normal touch-recognition algorithms. Another problem about collaborative displays is the orientation of the concertedly used resources. When multiple users are all standing behind the same edge of the screen, the orientation of the content is straightforward, however, as soon as people are spread around the surface, it is a difficult task to determine the direction in which to display objects on it. For example, Scott et al. [2003], Rogers et al. [2004] and Tang et al. [2006] all present papers addressing the collaborative issue. To avoid all these troubles, my application is designed to be used only by one single user in a first stage.

... however, Touch Graffle is designed for only one user

Definition: *Nodes, lines, text & objects*

NODES, LINES, TEXT & OBJECTS:
In order to use the same languague, I need to define a few terms.

- When talking about **nodes**, I mean the vertices of the graph.
- **Lines** will be any kind of connections between the nodes, no matter if they have arrows at the end or not.
- **Text** is obviously any label. They can be attached to a node, a line or simply exist on their own. I will also use the word ”label” as a synonym.
- If I talk about **objects**, it can be either a node, a line or text; anything that can be dragged around.

I used the Sprite Kit infrastructure

The application is written in Objective C, and uses the Sprite Kit infrastructure for rendering. According to Apple, ”Sprite Kit provides a graphics rendering and animation infrastructure that you can use to animate arbitrary textured images, or sprites” [1]. In order to recognize and process all the touches, I used the MultiTouchKit framework, a framework composed by my supervisor Simon Völker, which is currently only used for development purposes.

[1] https://developer.apple.com/library/mac/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html

Figure 5.1: The interface of the application is kept very simple. Only a thin grid and a ”graphics context” are visible when starting the program.

In the upcoming sections, I will present and explain the design decisions I took, followed by the gesture set I composed for this purpose.

5.1 The interface

The interface is deliberately kept simple. It just consists of a white background with thin horizontal and vertical lines. In the left bottom corner, I put an area which shows the current ”graphics context”. See section 5.1.2 for more details about this space. Figure 5.1 shows everything that is visible as soon as the program is launched.

The interface is kept simple

5.1.1 The background

The grid of the background is only added because of orientation purposes. As the interaction area is intended to represent a zoomable user interface (ZUI), the grid gives the user some visual feedback when zooming in or zooming out, especially when there is no content yet on the screen

The background is zoomable

that would scale too. Zooming can be achieved by pinching either four or five fingers on the background.

The selection area lets you select multiple objects

Furthermore, multiple objects can be selected by creating a so called selection area. When starting your trace on the background and moving it to any point, a rectangular selection area will be created, where the starting point and the releasing point will determine its diagonal. Every object that is completely within this area will be highlighted. This is different than standard graphics programs, where your selection area usually only needs to intersect an object in order to be selected. The reason why I changed this can be seen in Figure 5.2. The left upper corner of our selection area was the starting point of our trace. It needs to start on the background in order for the system to know that we want to do a selection. So imagine we would like to move the two orange hexagons on Figure 5.2 without changing the position of the outer white rectangle. With the standard selection method, our selection area would always intersect the white rectangle too, which results in the fact that we could only move the two hexagons separately. The drawback of this technique is that we need to create big selection areas when we plan to select multiple big objects.

By the way, all the selected objects will be highlighted in yellow, and the selection area will disappear as soon as the user releases the trace.

5.1.2 The current graphics context

On the left lower corner is the current graphics context

As mentioned before, the only visible "menu" is in the left bottom corner. However, as you could expect at first sight, it is not a classical menu that you know from typical desktop applications like Adobe's Illustrator for example. Actually, this area does not let you do much, it rather represents the current drawing parameters you are using when creating a new object.

In my pilot study, I realized that users tend to draw similar things directly after each other, sometimes even before dragging them to the intended places. Another possible

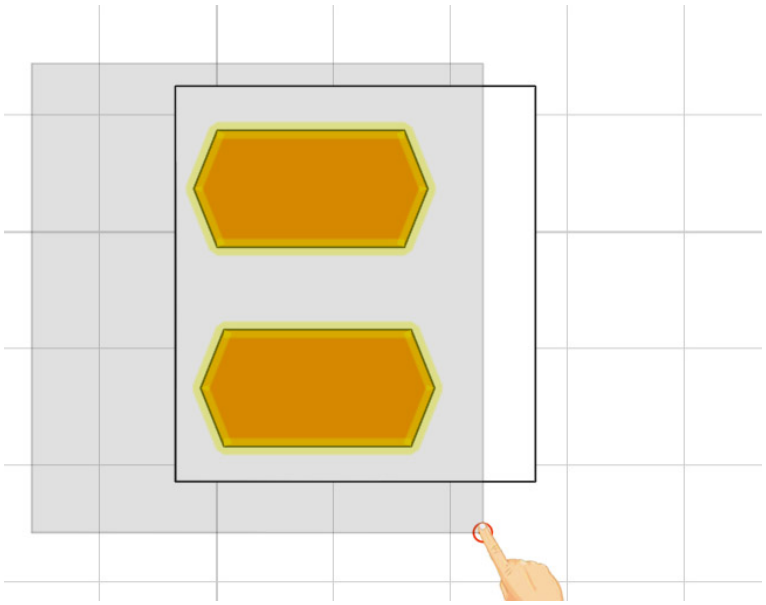

Figure 5.2: The selection area (grey) lets the user select multiple objets. The objects need to be fully included in the selection area to be selected.

workflow is to create one object and then copy it as often as it is needed. The graphics context is intended to save the users some time who do not use the copy-paste function. Instead of drawing a few standard rectangles and editing the properties of every single one of them, the current graphics context lets the user create one rectangle, edit its properties, and create all the following nodes with the same properties. The context is divided into three parts, one part for the nodes, one for the lines and one for the texts. This means that when you edit the stroke color of a node on your screen, only the border of the representative node will change its color, without affecting any of the other properties. Figure 5.3 shows the change of the graphics context as soon as we edit the fill color of a node. How to change properties of the different objects on the screen can be read in section 5.4.7 for nodes, in section 5.5 for lines and in section 5.6 for texts.

The graphics context is helpful when creating similar things consecutively

The only interactive part happens when you touch one of the three context items. When you tap on them, the sys-

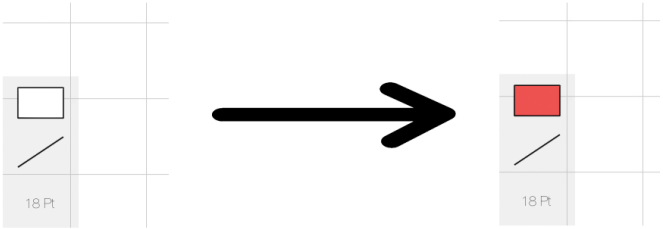

Figure 5.3: By changing the fill color of a node, the graphics context updates too, so that the next node can be created with the same properties as used before.

tem will jump back to the standard properties for this item, which are a black solid thin stroke color and a white fill color for the nodes, a black solid thin arrow-less stroke color for the lines and a black 18pt font for the texts.

5.2 The alternative trace

The alternative trace lets you ”hover” over objects

One big difference between mouse input and touch input is that the mouse can be active or not. It lets the user hover over certain areas without manipulating them directly. If she wants to perform an action on the current position, she just needs to click one of the mouse buttons.

Touch input is different because there is no difference between touching and hovering. Standard multi-touch surfaces do not support the detection of fingers above the surface yet. Han and Park [2012] and Takeoka et al. [2010] construct hover-based interfaces that let the user exploit the area above the surface. However, these interfaces are not a standard, and as long as this is not the case, we either have to create our own system which recognizes hovering, or we do a workaround. I chose the second option. One restriction is that the systems of Han and Park [2012] and Takeoka et al. [2010] could differentiate between different

Figure 5.4: A notification at the top of the screen leads the user to the next thing she can do.

levels above the surface, whereas mine could not. In order to activate this alternative trace, the user needs to hold down his finger for 0.5 seconds on approximately the same place. The virtual trace will change its color to blue and it will also scale twice as big, so the user knows she is currently in the alternative mode.

Activate the alternative trace by holding your finger down for 0.5 sec

The idea about an alternative touch is not new, Kin et al. [2011b] also invoke a special touch, however they use two fingers that are very close to each other. They call it "the conjoined touch". At first, I wanted to use the same concept, but I soon realized that this will be more difficult than expected. If the two touches are too far apart, the system thinks we want to create one of the two standard objects (Figure 5.5 (b1)&(b2)), and if they are too close together, the system thought they are one single one that is moved very quickly. This is amplified by the fact that the normal pinch gesture could not have been applied to small objects anymore, because the system would have thought it was an alternative trace. That is the reason why I chose the long press to invoke the alternative trace, even though one should avoid gestures that are based on a timer.

The idea is from another paper, where it is called "conjoined touch"

The alternative mode lets people move over objects without actually moving the object itself. This will be helpful in the section 5.7, where I will explain the context menus.

5.3 Notifications

Notifications provide helpful feedback

In order to help the user in different situations, I decided to give her some hints when I thought that she could be lost, or when I thought that she did not know what she could do next. In these cases, a grey area will appear at almost the top of the screen including a main text and a subtitle helping the user understand the current situation. Figure 5.4 shows a notification that appears when holding an alternative trace above a label.

Fading out a notification can be done manually or automatically

When showing a notification, we first have to cancel all the running actions that want to hide the notifications. As the notification is a singleton, it can happen that one class wants to hide its notification while another one wants to show a new one. In this case, the hiding needs to be canceled. One property that can be set is if we want the notification to hide automatically after a certain delay, or if we want to do it manually. Listing 5.1 shows the method that takes care of all the showing of notifications.

5.4 Nodes

The nodes are the vertices in the graphs. Right now, they can only be rectangles, ellipses, hexagons or stars. However, they could easily be extended by other graphical nodes or even images.

5.4.1 Creating nodes

When you plan to create a node, you are given 3 possibilities.

Use the ”node creatingcontext menu” to create a node

The first possibility is to create a new object by invoking the ”node context menu” (Figure 5.5 (a)). This one appears as soon as you tap somewhere on the background. In classical desktop applications, new nodes can normally be created from a side menu, but as we do not have one because of

Listing 5.1: Displaying a notification

```
- (void) showNotification:(NSString*)string
    withSubstring:(NSString*)substring
{
   _mainLabel.text = string;
   _subLabel.text = substring;
   [_bgSprite
       setSize:CGSizeMake(MAX(_mainLabel.frame.size.width,
       _subLabel.frame.size.width)+50, 80)];
   [NSObject cancelPreviousPerformRequestsWithTarget:self
       selector:@selector(hideNotification) object:nil];

   if (self.alpha == 0.0)
   {
      SKAction *fadeIn = [SKAction fadeInWithDuration:0.3];
      [self runAction:fadeIn completion:^{
         if (_hideNotificationAutomatically) {
            [self
                performSelector:@selector(hideNotification)
                withObject:NULL afterDelay:1.0];
         }
      }];
   }
   else
   {
      if (_hideNotificationAutomatically) {
         [self performSelector:@selector(hideNotification)
             withObject:NULL afterDelay:1.0];
      }
   }
}
```

dimension reasons, we will invoke this "node creating context menu". When we see our different options (which can of course be extended in the future), we just need to drag the object out. The new node will be hooked under your finger and the context menu will disappear again. While the menu is shown, I put a semi-transparent grey layer underneath it to take the focus from the rest of our scene.

Use the shortcuts to create a rectangle or an ellipse

In the graph study I found out that the rectangle and the ellipse are by far the most popular nodes. These two standard nodes can be created by moving two fingers on the background. The two points will represent 2 diagonal corners of a bounding box where a rectangle or an ellipse will be drawn inside. The kind of the node depends on the position of the right touch. If this one is above the left one, the system will create an ellipse as soon as one touch is released (Figure 5.5 (b2)), otherwise a rectangle will be created (Figure 5.5 (b1)).

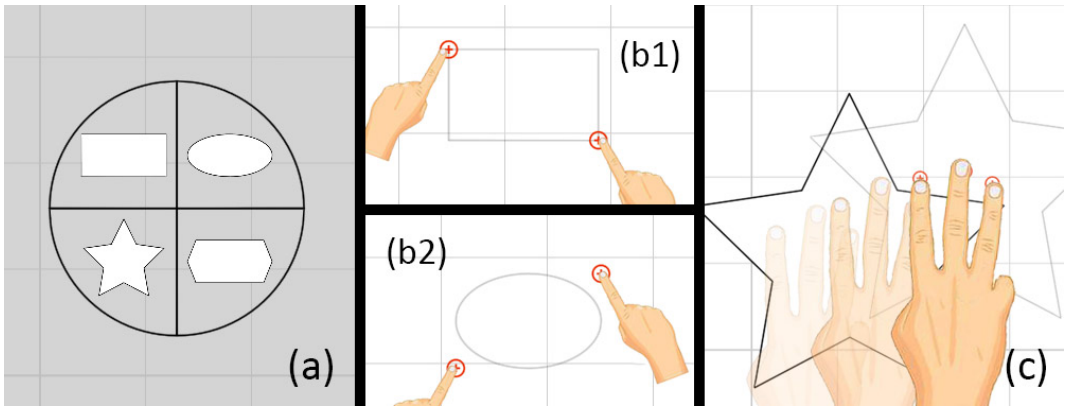

Figure 5.5: The three possibilities to create new nodes: (a) Create a node from the context menu. (b1) and (b2) show how to create the two standard nodes. (c) Copying a node.

Copy an existing node

The last possibility is to copy an existing node by using the 3-finger-dragging technique (Figure 5.5 (c)). A more detailed description can be found in section 5.4.3.

5.4.2 Manipulating a node

Use two fingers to rotate or scale a node

Manipulating a node is very straightforward. One finger can be used to drag the object, while two fingers are used to rotate and scale the object. The latter two manipulations work as you would expect, with one small exception. When scaling the object and your two fingers are aligned with the object, either vertically or horizontally, the object will also only scale in x- or y-direction. If you want to scale proportionally, the two touches should be aligned diagonally in relation to the object being edited. Listing 5.2 shows the code snippet of how a node can be scaled in one direction only. I compare the vector between the two touches with the vector of the object. If the vectors are parallel or orthogonal to the orientation of the node (including a small variance), only the *hScale* or the *vScale* variable is set. In the other case, both variables are set and the scaling is done proportionately.

Listing 5.2: Separate scaling of a node

```
float relativeScaleFactor = 1.0f;
float hScale = 1.0f;
float vScale = 1.0f;

//Get the angle between the 2 touching points and the
    rotation of the object modulo PI
float angleBetween2Points =
    atan((secondStartPoint.y-firstStartPoint.y)/
    (secondStartPoint.x-firstStartPoint.x));
float moduloAngle =
    fmod(self.zRotation-angleBetween2Points, (M_PI));

if(moduloAngle < 0)
{
   moduloAngle += M_PI;
}

//Check if the 2 touching points are aligned on one line
    +- scaleAngleOfDeviation (horizontal and vertical)
if(moduloAngle > M_PI-scaleAngleOfDeviation || moduloAngle
    < scaleAngleOfDeviation)
{
   hScale = relativeScaleFactor;
}
else if(moduloAngle > M_PI_2-scaleAngleOfDeviation &&
    moduloAngle < M_PI_2+scaleAngleOfDeviation)
{
   vScale = relativeScaleFactor;
}
else
{
   hScale = relativeScaleFactor;
   vScale = relativeScaleFactor;
}
```

5.4.3 Copying

Copying is done with three fingers

Copying can be done with three fingers, all heading in the same direction. Depending on the release of the semi-transparent temporary copied object (Figure 5.6 (a)), which is bound to your three fingers and which can be dragged around the screen, either a copy of the whole object will be created, or the properties of the starting object will be copied to the dropped object.

You can not copy lines without their bounded nodes

I decided to only be able to copy nodes, because lines should always be attached to a starting node and an ending node, and small labels are difficult to hit with three fingers. Furthermore, the same kind of lines and the same kind of

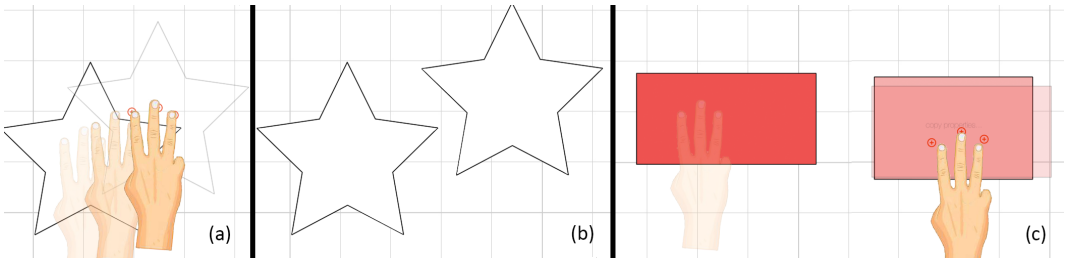

Figure 5.6: (a) A semi-transparent temporary copy is created when moving three fingers out of a node. (b) Releasing the temporary copy over the background will create a copy of the node. (c) While holding the temporary copy over another node, the latter one will start blinking, indicating that only the properties will be copied to this node.

texts can be created one after the other because of the ”current context” we are in.

Copying nodes

There exist two cases in which a node can be created.

In the first case, the temporary copied object that we are dragging around is released on the background. In this case an exact copy of the node, including its label, will appear on the screen (Figure 5.6 (b)).

The other option is that the temporary copied object is released on another object. In this case, I still check if the temporary object has been dragged and then released directly, or if it has been held over the other object. In the first case I will create a copy above the other object; in the second case, I will copy the properties.

Copying properties

You can also copy properties with the same gesture

Whenever the temporary copy of the object is held over another object, the latter one will start blinking after a certain time, and a text that says ”Copy properties...” will indicate that you will only copy the properties to the targeted object

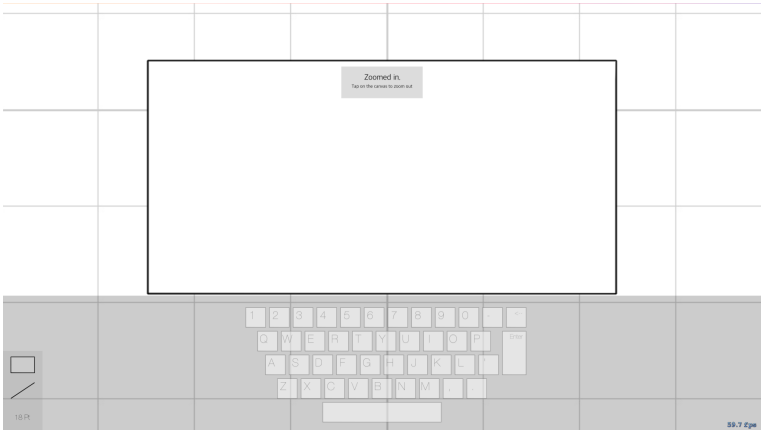

Figure 5.7: By double tapping on a node, we can add text to it.

when releasing the temporary copy (Figure 5.6 (c)).

5.4.4 Adding text

Text can be added to nodes by double tapping on them. The interface will automatically zoom to the node and a self-made keyboard will appear at the bottom of the screen (Figure 5.7). Also a notification will inform the user that she can quit editing by tapping somewhere on the background.

Double tap to add text

5.4.5 Editing the z-Position

By triple tapping on a node, this one can be brought to the front if it is hidden by another node. As lines are always attached to two nodes, their z-Position will be minimally smaller than the one of the node which is the furthest back. A gesture to send nodes to the back was not implemented.

Triple tap to edit the z-Position

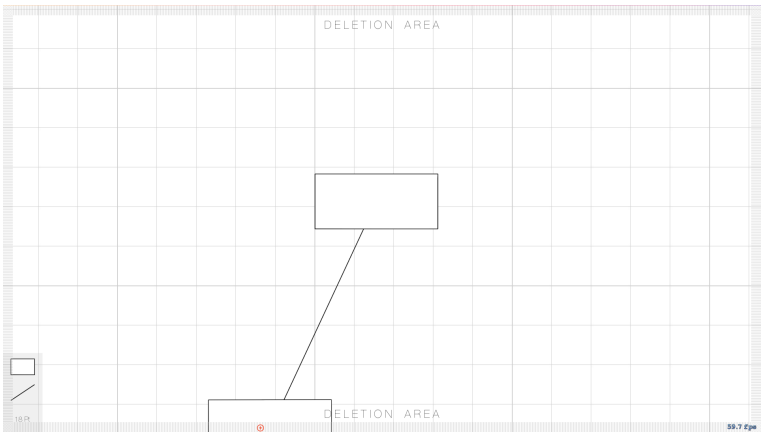

Figure 5.8: nodes can be deleted by dragging them to the edge of the screen. A dashed border will make you aware that you are currently in the deletion area.

5.4.6 Deleting a node

Delete a node by dragging it to an edge of the screen

Deleting a node can be done by dragging it to one of the screen edges. When moving a node to this area, a dashed border will appear on the edges indicating that you are currently in the deletion area and that the dragged node will be removed when releasing it. Together with the object, all the attached lines, that would have no end point anymore, will be removed too. Figure 5.8 shows the deletion area.

5.4.7 Editing the properties of a node

Make use of the alternative trace to get the context menu

When editing the properties of a node, we make use of the alternative trace explained in section 5.2. Depending weather we want to edit the border of the node, or the node itself, we move the alternative trace to the respective area, followed by tapping somewhere on the background where we want the context menu to appear. For example, if we want to change the border color of our node, we move the alternative trace to our node, so it intersects the border around it. Then we tap somewhere on the screen so that the corresponding context menu pops up where we select the color. Further manipulations can be found in section 5.7.

5.5 Lines

In my program, lines will always be attached to two nodes, they can have a few kinks, and they can be direct or indirect. In terms of properties, we can change the color, the width and the style (solid, dashed or dotted).

5.5.1 Creating lines

The task to figure out what the best way would be to create lines was not the easiest one. I asked a few colleagues how they would draw a line between 2 nodes, and the first answer I often got was to draw a line with one finger from the starting node to the end node. However, what they did not consider was that the system will think that you want to move your node with one finger, so that the starting node will just end up on top of the end node if you use this gesture. So my first thought was to add an area around the node, where the user could start her line. Nonetheless, I abandoned this idea again, because as it just took more screen space, some of these areas could overlap and it did not make much use of the freedom of multi-touch.

My solution now makes use of three fingers. Two fingers pin the starting node, while a third finger, logically from the other hand moves from this node to the end node, drawing a temporary line. As soon as you release your finger above the new node, the line will be definite. Figure 5.9 shows the sequence of this gesture.

Pin the starting node and draw a line with the other hand to the ending node

5.5.2 Modifying the path

When touching a line with one finger, and performing a gesture like you would drag it away, the two points around the touch, which can be endpoints of the line or kinks, will remain where they are. There will only be a point added to the line underneath the touch which you can be moved around. Theoretically, it is possible to create 2 new kinks

Touch the line and drag it away to create a kink

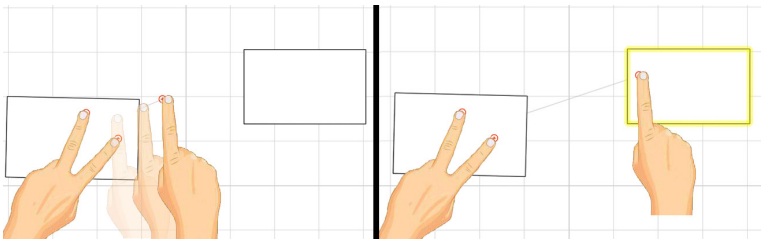

Figure 5.9: Create a line by pinning one node with two fingers and by dragging the line out of this node towards your end node. The end node will be highlighted when you reach it.

and move them to two new points simultaneously, however as people can only focus on one movement if they are not linked, this will probably more often than not be performed in a sequence.

5.5.3 Deleting

Delete a line by dragging a kink to the border of the screen

We can either delete a whole line, or only one of its kinks. Deleting the whole line is done the same way as deleting a node, by dragging one of the kinks to the edge of the screen so that the deletion area will appear. Once you release the kink in this area, the whole line will be removed.

Delete a kink by making use of the alternative trace

If the user wants to delete a kink, she has to make use of the alternative trace again. As soon as she moves it above a line, semi-transparent crosses will appear above each kink. Touching one of them will not delete the kink right away but the alpha of the cross will only change to non-transparent. When releasing the alternative trace, all the kinks with a non-transparent cross on top will be deleted.

I could have also deleted the kink directly when touching one of the crosses, however the line would then snap back and would probably not be placed underneath the alternative trace anymore. Then the user would need to reposition this trace again in order to continue modifying the line.

Figure 5.10 shows an example scenario. The blue trace rep-

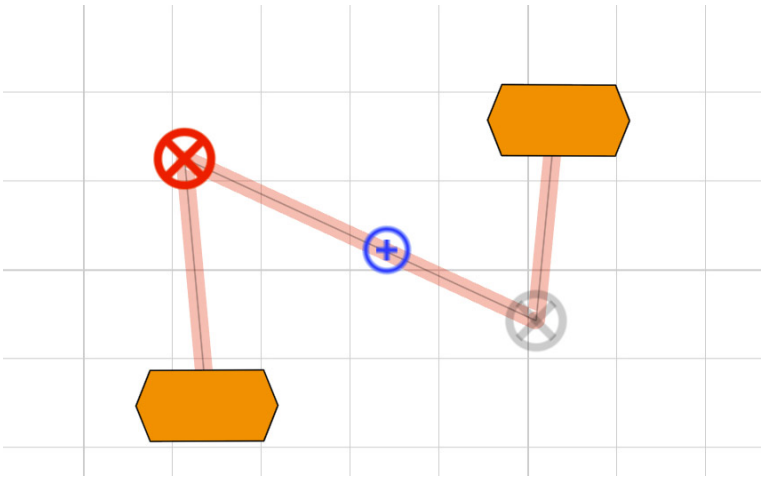

Figure 5.10: Move the blue alternative trace over the line to make the semi-transparent crosses appear. Touching one of these crosses will make them become non-transparent. Once the alternative trace is released, all the kinks with a solid red cross on top will be deleted.

resents the alternative trace that has been activated on top of the line. The latter one "recognizes" that the line has been selected and that the semi-transparent crosses will be added. In our case, the user already touched the upper left cross, wherefore it became non-transparent and red. Once the user releases the alternative trace, the kink with the red cross will be deleted and the resulting line will be composed of two segments only.

Add text by double tapping

Adding text works the same way as with nodes. By double tapping on the line, a virtual keyboard will be slid in, letting the user type whatever she likes.

The properties of a line can also be changed in the same way as changing the properties of a node. The context menu can be invoked by moving the alternative trace above the line, which gets highlighted, and by tapping somewhere on the background where you want the context menu to appear.

5.6 Texts

Texts can be attached to nodes, lines or the background

Texts can be attached to either a node, a line or to the background. Properties that can be changed are the orientation, the font-size and the color. As texts can be hard to grab when scaling or rotating them, I'm offering a second alternative to manipulate these properties.

When a label is attached to a node or to the background, we just need to define its relative position to its parent. When moving the node, Sprite Kit takes care of moving the children too. However, when we add text to a line and we add a kink or move a kink of the line, we actually do not move the frame of the line. In this case, we have to recalculate the position of the label manually, depending on the surrounding kinks, and the position between these ones. Listing 5.3 shows the function that has been implemented to recalculate the position of the label on a line. *pos* represents the absolute position where the label has been dropped.

Text is added by double tapping

Adding text can be done by double tapping anywhere on the surface. Depending on the location of the tap, the text will be either solitary or bound to a node respectively a line. In the last two cases, it will also adapt the transformations that are performed on its parent.

As I am using a zoomable interface, I also decided upon zooming to the edited object as soon as the double click is recognized. Listing 5.4 shows the calculations that needed to be done so that the node or the line could fit into the area above the appearing keyboard.

Delete text by dragging it to the border of the screen

Deleting text works again the same way as deleting nodes or lines. Drag the text to one edge of the screen and release it in the deletion area that appears.

The font size can be changed without a menu

In order to open the context menu, which in this case only lets you change the color of your text, the alternative trace needs to be moved on top of the text, followed by a touch on the background. Changing the font-size and the orientation can be either done with the normal two-finger-gestures or with the following alternative.

Listing 5.3: Recalculate the position of a label on a line

```
- (void) recalculateLinePercentageFromPosition:(CGPoint)pos
{
   for (int i = 0; i < [_pointArray count]-1; i++)
   {
      NSValue *value1 = _pointArray[i];
      CGPoint point1 = [value1 CGPointValue];

      NSValue *value2 = _pointArray[i+1];
      CGPoint point2 = [value2 CGPointValue];

      // Save the position in the specificPositions if the
          position of the label intersect the rectangle
          created by two consecutive kinks
      CGRect rect = CGRectMake(point1.x, point1.y,
          point2.x-point1.x, point2.y-point1.y);
      CGPoint labelPos = [self convertPoint:pos
          fromNode:[[MTKTable table] currentScene]];
      if (CGRectContainsPoint(rect, labelPos))
      {
         [_specificPositions setObject:[NSNumber
             numberWithInt:i] forKey:@"lastPointForLabel"];
         [_specificPositions setObject:[self
             getPercentageBetweenPoints:@[_pointArray[i],
             [NSValue valueWithCGPoint:labelPos],
             _pointArray[i+1]]]
             forKey:@"percentageForLabel"];
         return;
      }
   }
}
```

Listing 5.4: Zoom to an object when editing its label

```
- (void) zoomToObject
{
   CGRect accFrame = [self calculateAccumulatedFrame];

   float scaleFactor = MIN([[MTKTable table]
       currentScene].frame.size.width/(accFrame.size.width+200),
       [[MTKTable table]
       currentScene].frame.size.height/(accFrame.size.height+200));
   SKAction *moveAction = [SKAction
       moveTo:CGPointMake(-scaleFactor*self.position.x+[[MTKTable
       table] currentScene].size.width/2,
       -scaleFactor*self.position.y+[[MTKTable table]
       currentScene].size.height/2+100) duration:0.5];
   SKAction *scaleAction = [SKAction scaleTo:scaleFactor
       duration:0.5];

   SKAction *combinedAction = [SKAction
       group:@[moveAction,scaleAction]];

   SKSpriteNode *bg = [[GestureManager sharedManager]
       background];
   [bg runAction:combinedAction];
}
```

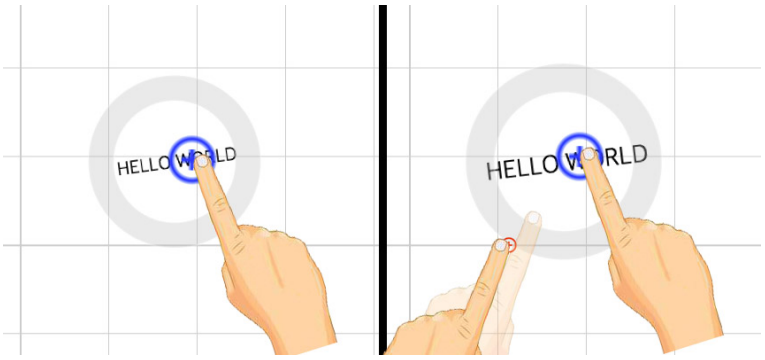

Figure 5.11: To activate the alternative for scaling and rotating text, hold the alternative trace over a label. The circle that appears around the text can be used to transform it.

I implemented an alternative to scale text that is too small

Obviously, when an object you would like to touch is too small, it will be hard to hit it with your finger, especially when you use multiple ones. Every object could be shrunk to a size where it is not touchable anymore with more than one finger, however while implementing my application, I realized that most of the time this was the case for labels as they were too small after resizing them. Therefore, I added this alternative, where only one finger needs to touch the text. Again, we make use of the alternative touch. When holding this one above a label, a wide circle will pop up around it. The circle represents now the area for the second touch, which is needed for the rotation and scaling input. Figure 5.11 shows the circle that lets you modify tiny labels.

5.7 Context menus

With gestural interfaces, users are rarely dragging their fingers across the screen

The context menus are essential for an easy workflow on big multi-touch tables. As Saffer [2008] mentions, the placement of targets such as menu items on the edges of the screen makes good sense on traditional input devices. As the user can not overshoot the target, the hit target becomes huge. With gestural interfaces, users are rarely dragging their fingers across the screen as they do with a cursor, instead, they will likely lift their fingers and place it on the new target. The advantage of context menus is now that

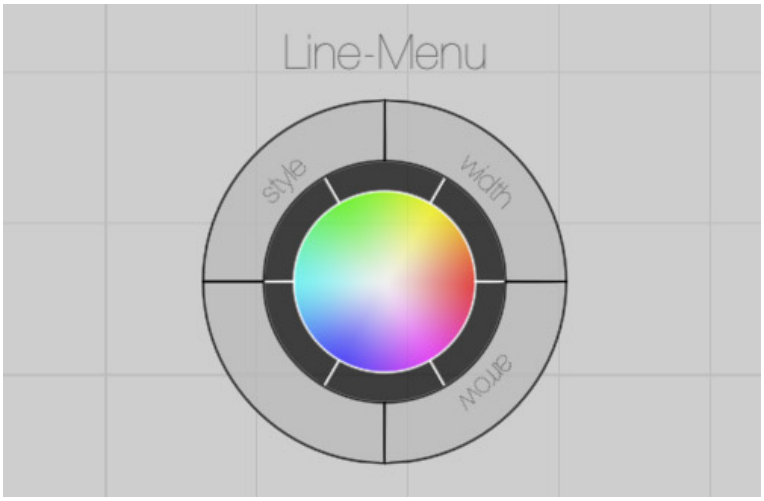

Figure 5.12: The context menu for a line. In the center we have a color picker, surrounded by the "color history" and further buttons.

they can be invoked directly at any point, without moving the hand to an edge of the screen. Furthermore, because of the proximity of the menu to the object, no big head swings are necessary.

To invoke the context menu, the alternative trace is moved over an object followed by a tap somewhere on the background. During the pilot study, I recognized that on the iPad, the drop-down menu was hiding almost one third of the screen when displayed, and a few users were complaining about the visual troubles they faced when changing the properties of an object that now got covered by the menu that popped up.

To avoid these overlaps, I decided to let the user decide herself where she wants to have the menu. There will also only be one menu shown at a time, so no connection will be needed between the object that is being edited and its corresponding context menu. Furthermore, everything except the edited object will be darkened (Figure 5.13 (b)), so the focus really lies on the latter one.

The user can decide herself where she wants the menu to appear

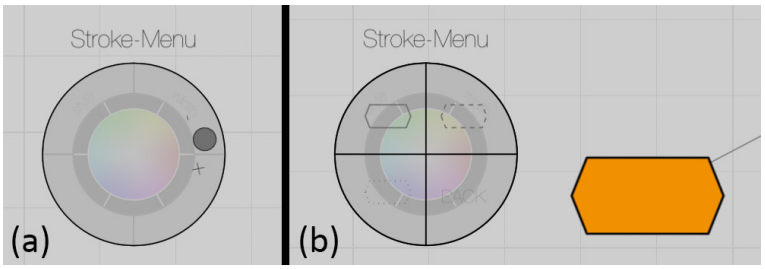

Figure 5.13: The context submenus. (a) A submenu for a larger number of options, in this case the choice of the line width. (b) A submenu for a small number of choices, in this case the choice of the stroke style that can either be solid, dashed or dotted.

The context menu consists of three circular areas

The context menus consist of three main parts. In the center is always a color picker circle. Around this circle, we have the ”color history ring”. This one includes the six last used colors, so that they are accessible directly without the need to select them again from the color picker. Around this ring, we have another ring with four buttons. Of course, the amount of buttons could be increased if needed, or even a new ring could be added, but so far four buttons were enough.

A preview of the stroke styles can be seen

Figure 5.12 shows an example context menu for a line. In this case, the buttons on the last ring let the user change the stroke style, the line thickness and the arrow direction. When changing properties that can only be in a few states, I chose a submenu like on Figure 5.13 (b), where the user can choose between three different stroke styles. The submenu then even shows a small preview of the different options. If the properties can have more values, I chose a turnable dial (Figure 5.13 (a)) as an input method.

So far, my work relies on some literature review, a pilot study and some own considerations, but in order to test its usability and intuitiveness, I still had to evaluate it by doing a second user study. The procedure and the results will be communicated in the next chapter.

Chapter 6

Evaluation

Based on the DIA-cycle, the designer should always iteratively design, implement and analyze her project. In this paper, the design and the implementation have been discussed in the previous chapter, while the analysis has not been done yet. One could say that the pilot study has been the analysis of the first iteration, but nevertheless I still need to evaluate my own work.

6.1 Participants

I had to replace two of my volunteers from the pilot study

For the participants, I tried to use the same ones that I used three months before for my pilot study. However, two of them were not able to help me out anymore, so I looked for two new ones. Both volunteers I asked, one male and one female, that also fitted in the same age category and the same field of study as the two leavers, directly agreed to act as a stand-in. As the only left-handed person did not participate anymore, and as the two replacements were both right-handed, the group only consisted of right-handed people. In a small form they had to fill out, I figured out that all of them use a smartphone everyday, six use a tablet sometimes (one every day, and one never) and four never used a tabletop, three a few times and one person uses it everyday.

6.2 Procedure

I recorded the user again for later reviews

The first thing the user had to do when entering the closed environment, where the study was held, was to sign a consent form, so I was able to record her hand movements using a GoPro. After that, I gave a short explanation about the goal of the study, I told them that I wrote the software on my own, and that there could still be bugs in the code which could lead to a crash of the system. I told them that they should not be nervous and scared to do any mistake.

The users could look at a cheat sheet when they forgot a gesture

Thereafter, I used a self-made cheat sheet, to explain all the gestures to the users and told them afterwards that I would lay it just behind them on a table, so that if they would forget a gesture, they could always turn around and look it up again.

Finally, the user had to recreate the two graphs from the pilot study (Figure 4.1). All of the six people that participated in the earlier study, claimed that they remembered doing these graphs, but were not able to recall any details of them, so that any learning effects could be excluded. Figure 6.1 shows a user who is currently adding a line from one node to another one.

At the end, the users filled out a questionnaire

At the end, the user had to fill out one more questionnaire where she had to rate five gestures and point out what she would have preferred regarding the interface or the gestures. A small present was also handed to them to reward them for the time they sacrificed.

6.3 Results

Again, like in the pilot study, I will distinguish between the objective observations I did, the subjective feedback I gathered and the strategies that have been pursued.

Many concerns can be solved when the user gets more familiar with the software

In my opinion, many of the concerns that I could observe or that the people wrote down in the questionnaires would be solved over time, when the user gets more familiar with

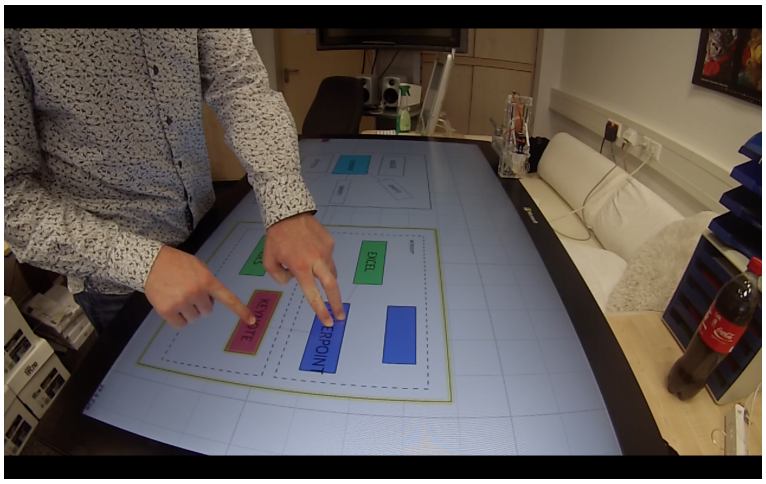

Figure 6.1: A user drawing an arrow in the C-Graph.

the program on the one hand, and the tabletop itself on the other hand. But as the system should also be handy for novices, I will point out every difficulty they faced.

In the upcoming chapter, I will present a few solutions to the issues I discovered, that should be improved in the future.

6.3.1 Observations

Beginners mix up things:
While the users, that use designing tools regularly, have a clear structure in their workflow, novices did everything promiscuously. The explanation for the experts is obvious, they probably map the workflow from their desktop computer onto the tabletop. Regarding the beginners, I think that the absence of modes (”pointing-mode”, ”draw-shapes-mode”, etc.), leads them to do whatever comes into their mind. The users do not need to click a button to come into an extra ”draw-lines-mode”, which can fasten up their working process.

Beginners did everything promiscuously

Current context is misleading for users:
The current context is probably only helpful when learned

over time. Two users knew the concept from other drawing tools, and had no problem adapting to it, however they were sometimes surprised by the outcome because they did not take care of it. They directly understood the situation and were able to adapt to it within seconds.

Beginners had difficulties with the current context

The newbies on the other hand, were often lost. After they realized why their shape had different properties than they expected, two were hoping that this shape would change to the standard shape when tapping on the current context menu. This however only set the context back, but not their shape. Furthermore, most participants did not comply with the current context. When they needed two similar shapes, they created the first one, changed the properties and then just copied it.

When touching a node, people expected it to be highlighted, so that you know it is active

Expected highlighting when touching node:
All the users that never used a tabletop before, were expecting the object they touch to change its appearance. This is for sure related to the desktop behaviour, where active objects have a bounding box around them so the designer has some handles that she can use to scale the object. On the tabletop however, these handles are superfluous because an object does not need to be active to be scalable, it is active all the time. As the users were expecting some kind of feedback, they held down their finger on the object, but then unintentionally activated the alternative trace.

Moving only nodes without the text inside is hard

Unintentionally moved text in nodes:
I offered the users the possibility to move, scale and rotate text with the same gestures they use to manipulate the containing node. Unfortunately, whenever a user wanted to move one of these nodes, she sometimes unintentionally dragged the text outside the node.

The timeout for copying properties was often activated unintentionally

Copy properties instead of object:
Another lack of feedback can be noticed when users try to copy objects on top of other objects. The interval, where they had to release the node in order to create a new one and not only copy the properties to the underlying one, was around one second. When the participants copied a node to another position, they often made sure that they moved it to the right position by looking at the template. In the

meantime, they held their fingers still on one point, and as soon as they reassured they were on the right spot, they released them. This often took more than one second, so that they copied the properties to the underlying node, and wondered why and where their copy disappeared.

Acrobatic wrenches when adding lines:
The banana subgraph is a good example for this behaviour. When adding the arrows to the subnodes, the user normally started on the left, therefore, she put the two fingers of the right hand on the top node and used the left index to draw the line. Then they repeated this for the middle and the right subnode. Five of my participants actually switched hands in between, because it was more comfortable, while the other three preferred leaving their double touch on the upper node and then had to cross their arms when drawing a line to the most right subnode.

Users did not always choose the most comfortable gesture

Females did not care about the arrow direction when adding lines:
I do not know if this is coincidence, but when the female participants only had to draw one line, they drew it in a way that they occluded the least with their hands and that it was the most comfortable for them, no matter in what direction the arrow of the line pointed. By using the context menu they then switched the arrow around. The males always pinned the starting object and drew the line in the same direction the arrow was heading.

female participants drew the line in the most comfortable way

6.3.2 Feedback

In my final questionnaire I handed to the users, I asked them to rate, on a scale from 1 to 5, the handiness and the remembering of some of the gestures. These were the gestures about copying nodes, copying properties, adding lines, deleting lines and deleting the kink of lines. As no one actually used the last gesture, and people only ticked the value how they imagined the gesture to be, I will leave this one aside. Figure 6.2 shows the remebering- and the hanidness-score of these four gestures.

Users had to rate a few gestures

What can be extracted from this graph is that the deletion

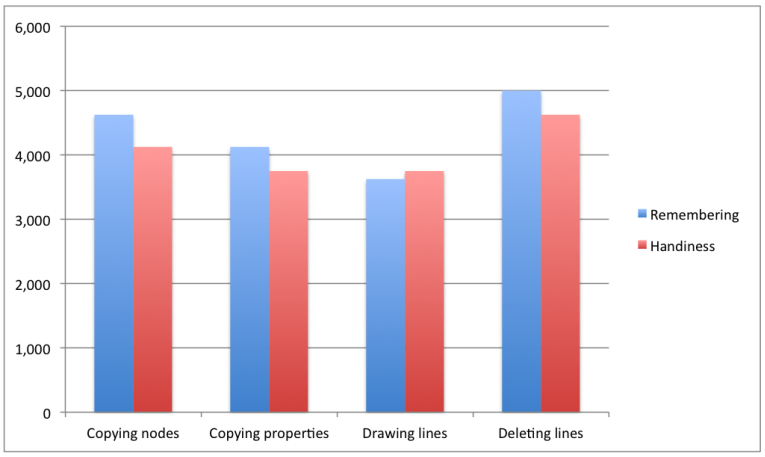

Figure 6.2: The score of the four gestures that I asked the participants to rate. Blue represents how well the gesture could be remembered and red represents how well the gesture could be performed.

Deleting lines is very easy

of lines was easy to remember and got a perfect score. Also most of the participants agreed that the way how to perform it was also relatively easy.

Copying properties is unhandy

The low handiness of the "copy properties" gesture is due to the fact I already explained in the previous section. Sometimes, when copying a node and holding it too long above another node, the latter one just adopted the properties of the copied one. This was often confusing for the participants.

Drawing lines came off worst

Drawing lines was the gesture I did the most reflections on in advance. At the end I was satisfied with the gesture, but as it seems, the participants still had the most problems with it. It is the only gesture where they found it harder to remember it than to perform it.

After that, I asked the participants to rate again on a scale from 1 to 5, if they would have preferred to have a gesture that could send something to the back, if they would have preferred one single menu for the stroke and the fill of nodes and if they would have preferred starting with a standard object instead of an object that is based on the current context. For the two last points, the score was an exact

3, so the users could not really agree if it was a good feature or a bad one. Only for the send-to-back feature, the result was more clear, as it got a score of 4, which means that most of the participants would have preferred to have such a feature.

Other concerns

At the end, I offered all my users the possibility to express all their remaining concerns down about the software. The worries that have been mentioned the most, are the following ones:

Undo:
Every person is so used to undo something with a single click, that they would have preferred to also be able to do it in my application. Because of technical reasons however, this was not possible, but it should for sure be implemented in the future.

Snaplines:
Snaplines are the lines that help you align certain nodes against other ones. They are very helpful if you want your graph to look more polished.

Snap back the pinned node when drawing a line:
When planing to draw a line, it is almost impossible to pin a node without also scaling it. If the system recognizes that you want to draw a node, it should lock the pinned shape, and transform it back to its original appearance as soon as the line gesture is recognized.

Lock text in a node:
To prevent the unintentional dragging of a text inside a node when you planned to move only the node, some kind of lock mechanism could be implemented to avoid this error.

Trigger the create menu in another way:
Each user, at one point, triggered the create menu accidentally when they touched the surface intentionally or unintentionally. It can be quickly dismissed again by clicking somewhere on the background, but it is still annoying. Maybe a slightly more complex gesture could solve the problem.

Layers:
One person also mentioned to use layers, that can be hidden or locked, so you do not mess up some finished parts.

6.3.3 Strategies

Only small changes in the strategies

Just like in the pilot study, I will look at the different strategies of the users, and I will try to identify if there are some disparities between then and now.

People focused more on subgraphs

Subgraph after subgraph:
This time, only one user created all her shapes first before concentrating on the details, and this only for the TF-Graph. The other ones always picked a subgraph that they finished first before concentrating on the next one. For the subgraphs, all imaginable strategies could be perceived.

Furthermore, the participant who used to first create the superclasses in the TF-Graph, did the same thing on the tabletop.

Left to right, back to front but less top to bottom:
In general, one can say that seven of the eight participants followed the same positional strategies than in the pilot study. However in the C-Graph for example, 3/7 of them did not create the upper ellipse at the beginning (in comparison to 8/8 and 3/4 on the iMac respectively on the iPad). For the different subgraphs, the top to bottom, left to right and front to back strategy was still dominant.

One user started all her graphs and subgraphs at the bottom

The one user who fell out of alignment started all her graphs from the bottom. In the C-Graph, she started with the Joystick-node, while in the TF-Graph, after completing the lower left ellipse, she continued with the Apple-node, that is situated in the lower right corner. She cut her way through from the lower subgraphs to the upper ones, applying in each the top to bottom, left to right and back to front strategy.

Text before colors:
Five of the eight participants always added the text before

the color, while the other three did a mixture. Nobody first added the colors followed by the text for the complete graph. People who did the text first, either added the colors directly after that, at the end of the subgraph or at the very end of the graph, but here no real strategy could be perceived.

Everybody added the text before the colors

Connections not always at the end:
Noticeable was that three participants changed their connection strategy. While on the iMac and on the iPad, they always waited until the end of a subgraph or until the end of the whole graph to add the lines, they added them whenever they wanted to do so. This certainly correlates with the absence of different modes, especially a separate line mode.

Connections were done in between

6.3.4 Speed

To compare the time the participants needed to complete some tasks, I chose a part of a graph that everybody finished. I settled for the left big container of the C-Graph, including all its subnodes and lines.

On average, the manipulation of Omnigraffle on the iMac was 46 seconds faster than the manipulation on the iPad and 1 minute and 11 seconds faster than the interaction with my software on the tabletop. As I was interested in seeing where these time differences came from, I decided to take a closer look at the detailed working process. I distinguished between:

In general, the iMac is the fastest and the tabletop is the slowest

- The time spent to create nodes. This included the positioning, the rotating and scaling of them.
- The time spent to change the properties of nodes. Changing the properties of lines and text was included in the respective lines and text parts.
- The time spent to add and edit lines.
- The time spent to add and edit text.

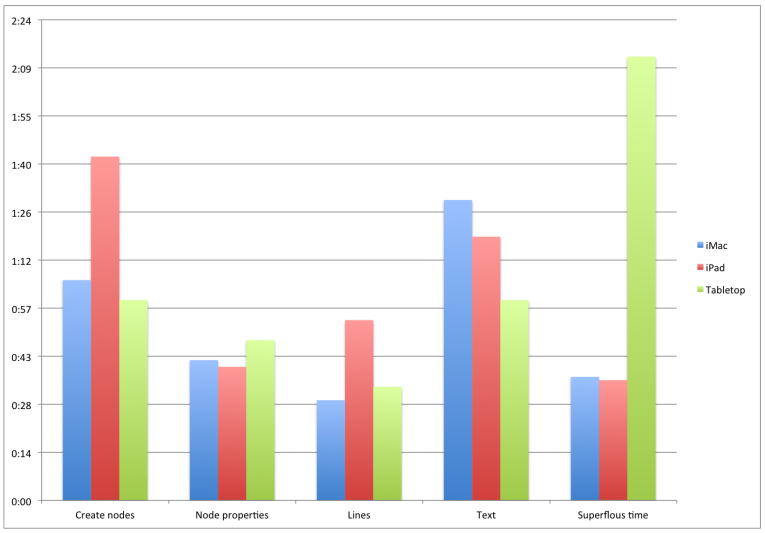

Figure 6.3: The average durations the users needed to recreate the different parts of the graph.

- The superfluous time. This time usually had two origins, the first one is the cognitive origin, where the user needed to think either about the gestures or her further approach. This also includes the time she needed to turn around and look at the cheat sheet I provided during the study. The other one is the destructive origin. When the users performed a wrong gesture, they had to rearrange their graph again if they destroyed something in it.

As I only had four people performing the C-Graph on the iMac, and the other four people performing it on the iPad, I also chose four random people where I analyzed the different performing times on the tabletop. The results of the investigation can be found in numerical form in Table 6.1 and graphical form in Figure 6.3

One user added the text separately and dragged it to the nodes afterwards

Surprisingly, adding and editing text on the iMac was the slowest. However, this was mainly due to one user, who needed over two minutes to add text. The reason therefore was that she always added the text separately on the background and dragged it to the nodes afterwards.

Task	iMac	iPad	Tabletop
Create nodes	1:06	1:43	1:00
Node properties	0:42	0:40	0:48
Add & edit lines	0:30	0:54	0:34
Add & edit text	1:30	1:19	1:00
Superfluous time	0:36	0:36	2:13
Constructive time	3:50	4:37	3:24
Superfluous time	0:36	0:36	2:14
Total	4:26	5:13	5:38

Table 6.1: The time needed for the different tasks while creating the lower left container node of the C-Graph

On the iPad, everything that had to do with creating objects took longer. Without the user, who I described in the previous paragraph, the processing of text would have probably taken longer on the iPad too.

On the tabletop, people actually needed the least real working time. They lost over two minutes either thinking about how to perform their next gesture, or rearranging the graph they just messed up by using a wrong gesture. Note that my software did not have an undo button, which could have saved a lot of time in the last case. What I also need to mention is that the menus of my program were very minimalistic, and only included the needed buttons, while the menus of Omnigraffle were more complex and needed more user interaction to achieve the intended result.

On the tabletop, people needed the least real working time

But all in all, one can say that the potential to quicken the creation of graphs on a tabletop is there. With a few more software improvements and people getting more used to multi-touch systems, especially big tabletops, creating graphs on these big interactive surfaces could be more intuitive, easier and faster.

The potential to quicken the creation of graphs on tabletops is existing

6.3.5 Conclusion

As mentioned in section 6.3.4, the actual construction time on the tabletop is shorter than on the other two devices. The

right multi-touch gestures combined with an attuned interface grants a faster manipulation on multi-touch interfaces.

However, there are still a few barriers that need to be broken down until tabletops can replace the desktop computer in terms of effectiveness and efficiency. It is hard to change the operating principles of people who are used to work with a mouse, a physical keyboard and a screen that is oriented in a vertical way. But as soon as the community gets more familiar with direct-touch interfaces, and especially multi-touch interfaces, the interaction on these kind of screens will be more natural and therefore also faster.

Users need to get rid of the "mouse-and-keyboard thinking" before they can really exploit the advantages of multi-touch systems

Based on this premise, and with some improvements of the current product (undo, lock layers, snaplines, ...), I am confident that this could replace and improve the current desktop versions of graph creating tools in the future.

Chapter 7

Summary and future work

In the previous chapters, I explained the process of how I started with the literature research, took a closer look at existing graphs and did a pilot study to find out how people interact with software tools that let you create graphical images that mainly contain nodes and edges. In my own work, I implemented Touch Graffle, a multi-touch diagramming tool that takes care of all the details I gathered in the previous chapters. In the end, I evaluated my work by comparing it to the outcomes of the pilot study. This last chapter gives an overview of my whole thesis, and will also point out a few improvements that should be taken care of in order to further explore the creation of graphs on tabletops.

7.1 Summary and contributions

No matter if you are a computer scientist, a politician, a journalist or pursue any other kind of intellectual profession, you will always stumble across graphs. Additionally, as big tabletop screens become more and more popular, it's on the dice to investigate in more detail the creation of graphical images on these kind of surfaces.

You will always stumble across graphs

The main contribution was to create an expressive and non-overlapping gesture set that lets you create graphs

In this paper, the main focus lies on the implementation of a gesture set, that can be used to create diagrams on big multi-touch tables. This set, that is not supposed to clash with itself, should be easy enough to perform and to remember. The adoption of multiple direct input points can not only quicken the tasks, but it can also let the user focus on what she wants to do and not on how she needs to do it. Furthermore, I identified different strategies depending on the device the user is working on. In a subgraph, people create their objects from the top to the bottom, left to right and back to front. The only difference can be perceived when considering the whole graph, because on big surfaces they move towards creating the subgraphs with closer proximity first. The last benefit of this paper was to see how a person gets along the absence of any kind of visible menu for these kind of programs. Only context menus can be popped up wherever the user wants to do so. As no participant in my later study complained about it, I assume that this is the right choice for big direct-touch surfaces.

A final user study helped me find out if the concepts were well grasped

After the implementation of my own program, which was derived from the initial literature research and the previous pilot study, I did a final user study in order to see how well this new concept is grasped. I did some own observations, where I realized that many users try to map their habits from the computer, that they are very familiar with, to the tabletop and expect the latter one to respond in that way too. A comparison of the strategies showed that people process subgraph after subgraph as this is more overseeable than to consider the graph as a whole. When focussing on the completion times, one can see that the tabletop comes off worst. However, after a more detailed investigation, the reason for this loss of time is the constant recall of the available gestures. Also the possibility to undo stuff and to lock some objects, could have quickened the whole process.

Further perceptions and their potential solutions will be amplified in the upcoming section.

7.2 Future work

In order to optimize the creation of diagrams on multi-touch tables using bimanual input, there are some details to be taken care of when examining it in the future.

First of all, there is no getting around a long-term study, where users will be using the software on a regular basis. This way, the constant remembering of the gestures should be minimized and the work of the end user should become more fluid.

A long-term study in the future will be necessary

Right now, the software is only designed for one single person standing at one fixed border of the tabletop. However, big multi-touch tables encourage the simultaneous handling of multiple people, scattered all around the table. There is a lot of research that has been done in this area, which should absolutely be considered first before expanding the software to handle multiple users.

Collaborative working should also be considered

Last but not least, I will again point out the most important enhancements that need to be done first. Every future developer, who will draw on my project, should first consider fixing the following issues.

Users seemed to have the most problems with the gesture that lets them create lines. If after a long-term study, this issue is still present, another more natural gesture should be considered for this task.

The create-line-gesture should be reconsidered

One does not get around an undo button or gesture. Every person using a program expects to easily return to a previous state. Like this, people are more open to try new things and are not afraid to do a mistake that may not be reverted.

Undo is a must

When using snaplines, the working time of a user can be accelerated because she is not fiddling around with some small resizing gestures to make the graph look polished.

Snaplines can be helpful

One could consider working with layers, that can be locked or hidden. This way, certain parts of the graph that are finished can not be altered by wrong gestures, like for exam-

Layers could also be considered

ple the inner text of nodes, that has been often edited unintentionally. Hiding the layers may help to get a better overview and to focus on the parts that are currently important.

If using the current context, it should be more noticeable

In my opinion, the idea of the current context is not that bad. However, it should be placed more central, or there should be more notifications where the user is reminded that it still exists. Being able to change the context manually, or to go back to a previous context, could also constitute an advantage.

The copy gesture was easy, and used a lot, however, it often transitioned into the unintended copy properties gesture

The copy gesture was a gesture that was used a lot, but often it ended up in unexpected results, so that a revision of it is a must. Using timeouts is always a less-than-ideal solution, therefore, another more distinguishable possibility to transfer the properties from one object to another one should be taken into account.

All in all, when taking care of all these trifles, I am optimistic that this project can help with the creation of diagrams on multi-touch tables in terms of speed, intuitiveness and straightforwardness.

Appendix A

Recreation times

In order to compare the times the users needed to recreate a certain part of a graph, I kept track of the times they spent. As a part of a graph, I decided on the left lower big container of the C-Graph. For the times, I differentiate between the time to create node, the time to change the property of nodes, the time to add lines, the time to add text and the remaining superfluous time.

The first four tables represent the times on the iMac, the next four represent the times on the iPad, and the last four represent the times on the tabletop.

Task	Time
Create nodes	1:16
Node properties	0:47
Add & edit lines	1:10
Add & edit text	1:26
Superfluous time	0:51
Constructive time	4:39
Superfluous time	0:51
Total	5:30

User M1

Task	Time
Create nodes	0:57
Node properties	0:21
Add & edit lines	0:28
Add & edit text	1:50
Superfluous time	0:24
Constructive time	3:36
Superfluous time	0:24
Total	4:00

User M2

Task	Time
Create nodes	1:03
Node properties	1:01
Add & edit lines	0:12
Add & edit text	0:42
Superfluous time	0:17
Constructive time	2:58
Superfluous time	0:17
Total	3:15

User M3

Task	Time
Create nodes	1:10
Node properties	0:39
Add & edit lines	0:15
Add & edit text	2:05
Superfluous time	0:56
Constructive time	4:19
Superfluous time	0:56
Total	5:15

User M4

Task	Time
Create nodes	1:35
Node properties	0:45
Add & edit lines	0:20
Add & edit text	1:10
Superfluous time	0:30
Constructive time	3:50
Superfluous time	0:30
Total	4:20

User P1

Task	Time
Create nodes	2:20
Node properties	1:05
Add & edit lines	1:10
Add & edit text	1:50
Superfluous time	0:35
Constructive time	6:25
Superfluous time	0:35
Total	7:00

User P2

Task	Time
Create nodes	1:27
Node properties	0:35
Add & edit lines	1:02
Add & edit text	0:55
Superfluous time	0:21
Constructive time	3:59
Superfluous time	0:21
Total	4:20

User P3

Task	Time
Create nodes	1:33
Node properties	0:15
Add & edit lines	1:03
Add & edit text	1:25
Superfluous time	0:59
Constructive time	4:16
Superfluous time	0:59
Total	5:15

User P4

Task	Time
Create nodes	1:28
Node properties	0:57
Add & edit lines	0:40
Add & edit text	0:55
Superfluous time	4:25
Constructive time	4:00
Superfluous time	4:25
Total	8:25

User T1

Task	Time
Create nodes	0:58
Node properties	1:35
Add & edit lines	0:39
Add & edit text	1:20
Superfluous time	1:08
Constructive time	3:32
Superfluous time	1:08
Total	4:40

User T2

Task	Time
Create nodes	0:36
Node properties	0:48
Add & edit lines	0:36
Add & edit text	0:59
Superfluous time	2:01
Constructive time	2:59
Superfluous time	2:01
Total	5:00

User T3

Task	Time
Create nodes	1:01
Node properties	0:53
Add & edit lines	0:24
Add & edit text	0:49
Superfluous time	1:23
Constructive time	3:07
Superfluous time	1:23
Total	4:30

User T4

Appendix B

The Gesture Set Cheat Sheet

The following two pages show the cheat sheet of the gesture set I defined. During the evaluation study, this cheat sheet, that I explained to the user in the forefront, was placed behind her so that she could always turn around to recall how to perform a specific gesture.

Gesture Set

NODES

CREATE A NODE	- **tap** on the screen to activate the shape menu OR - **pinch** on the background (**diagonal direction matters**!!)
MOVE A NODE	- place **1 or 2 fingers** on the node and move them to the position you want
SCALE AND ROTATE A NODE PROPORTINALLY	- place **2 fingers** on the node and pinch/rotate them
SCALE NODE IN X- OR Y-DIRECTION ONLY	- place your 2 fingers on the node so that they are **aligned** either **vertically or horizontally** with the object, and start pinching
CHANGE THE NODE PROPERTIES	- activate the "alternative trace" by **holding your finger down** until trace becomes blue - move the trace to either the **node itself** or the **contour** - **tap** somewhere on the **background** to activate the menu
COPY A NODE	- place **3 fingers** on the node and drag them to the new position
COPY THE NODE PROPERTIES	- place **3 fingers** on the node and drag them to the target node - as soon as the target node **starts blinking**, you can release
ADD TEXT TO A NODE	- **double click** on the node
BRING NODE TO FRONT	- **triple click** on the node
DELETE A NODE	- **drag the node to one of the edges** of your screen

LINES

CREATE A LINE	- put **two fingers** of one hand **on a node** - **drag another finger** from the other hand **to the target node**
CHANGE THE LINE PROPERTIES	- activate the "alternative trace" by **holding your finger down** until trace becomes blue - **move the trace to the line** - **tap** somewhere on the **background** to activate the menu
ADD TEXT TO LINE	**- double click** on line

MODIFY THE LINE PATH	**- touch the line** to create a new corner and **drag it** to the position you like
DELETE LINE CORNERS	- activate the "alternative trace" by **holding your finger down** until trace becomes blue - **move the trace to the line** - **tap on the "X"es** that appear on the corners
DELETE A LINE	- **drag the line to one of the edges** of your screen

TEXT

ADD TEXT	- **double click**
CHANGE THE TEXT PROPERTIES	- activate the "alternative trace" by **holding your finger down** until trace becomes blue - **move the trace to the text** - **tap** somewhere on the **background** to activate the menu
MOVE TEXT	- place **1 finger** on the text and move it to the position you want
SCALE AND ROTATE THE TEXT	- place **2 fingers** on the text and pinch/rotate them OR - activate the "alternative trace" by **holding your finger down** until trace becomes blue - **move the trace to the text** - use the **circle** that appears **around the text**
DELETE TEXT	- **drag the text to one of the edges** of your screen

OTHER

ZOOM IN/OUT	- **pinch 4 or 5 fingers** on background
SET CURRENT CONTEXT TO DEFAULTS	- **tap** on either the shape, the line or the text in the **lower left corner**
SELECT MULTIPLE NODES/LINES	- **drag with one finger** over the objects, starting on the background

Bibliography

Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. Medusa: a proximity-aware multi-touch tabletop. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 337–346. ACM, 2011.

R.M. Baecker. *Readings in Human-computer Interaction: Toward the Year 2000*. Interactive Technologies Series. Morgan Kaufmann Publishers, 1995. ISBN 9781558602465. URL `http://books.google.lu/books?id=gjm6FpMUTXgC`.

Gilles Bailly, Eric Lecolinet, and Yves Guiard. Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 591–594. ACM, 2010.

Ravin Balakrishnan and Ken Hinckley. Symmetric bimanual interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 33–40. ACM, 2000.

Nikola Banovic, Frank Chun Yat Li, David Dearman, Koji Yatani, and Khai N Truong. Design of unimanual multi-finger pie menu interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 120–129. ACM, 2011.

Xiaojun Bi, Yang Li, and Shumin Zhai. Ffitts law: modeling finger touch with fitts' law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1363–1372. ACM, 2013.

Eric A Bier, Maureen C Stone, Ken Pier, William Buxton, and Tony D DeRose. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80. ACM, 1993.

Peter Brandl, Jakob Leitner, Thomas Seifried, Michael Haller, Bernard Doray, and Paul To. Occlusion-aware menu design for digital tabletops. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 3223–3228. ACM, 2009.

William Buxton and Brad Myers. A study in two-handed input. In *ACM SIGCHI Bulletin*, volume 17, pages 321–326. ACM, 1986.

Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 95–100. ACM, 1988.

Xiang Cao, Andrew D Wilson, Ravin Balakrishnan, Ken Hinckley, and Scott E Hudson. Shapetouch: Leveraging contact shape on interactive surfaces. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 129–136. IEEE, 2008.

Maarten Cardinaels, Karel Frederix, Johan Nulens, Dieter Van Rijsselbergen, Maarten Verwaest, and Philippe Bekaert. A multi-touch 3d set modeler for drama production. 2008.

Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226. ACM, 2001.

Felix. Extremely efficient menu selection: Marking menus for the flash platform. `http://www.betriebsraum.de/blog/2009/12/11/extremely-efficient-menu-selection-marking-menus-for-t` 2009. [Online; accessed 05-December-2014].

Cédric Foucault, Manfred Micaux, David Bonnet, and Michel Beaudouin-Lafon. Spad: a bimanual interaction

technique for productivity applications on multi-touch tablets. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 1879–1884. ACM, 2014.

Salvador Fuentes, J Alfredo Sánchez, Osvaldo Huerta, and Ofelia Cervantes. Innovatouch: a multi-touch framework to support gesture recognition for innovation activities. In *Proceedings of the Companion Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction*, pages 12–13. Brazilian Computer Society, 2011.

Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of motor behavior*, 19(4):486–517, 1987.

Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C Olson. Faster command selection on tablets with fasttap. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 2617–2626. ACM, 2014.

Charlotte Häger-Ross and Marc H Schieber. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. *The Journal of Neuroscience*, 20(22):8542–8550, 2000.

Seungju Han and Joonah Park. A study on touch & hover based interaction for zooming. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 2183–2188. ACM, 2012.

Ken Hinckley, Randy Pausch, Dennis Proffitt, James Patten, and Neal Kassell. Cooperative bimanual action. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 27–34. ACM, 1997.

Christian Holz and Patrick Baudisch. Fiberio: a touchscreen that senses fingerprints. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 41–50. ACM, 2013.

Jean-François Jégo, Alexis Paljic, and Philippe Fuchs. User-defined gestural interaction: A study on gesture memorization. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*, pages 7–10. IEEE, 2013.

Xiaodan Jiao, Hui Deng, and Feng Wang. An investigation of two-handed manipulation and related techniques in multi-touch interaction. In *Machine Vision and Human-Machine Interface (MVHI), 2010 International Conference on*, pages 565–568. IEEE, 2010.

JPowered. History of bar charts and graphs. `http://www.jpowered.com/graphs-and-charts/bar-chart-history.htm`. [Online; accessed 29-October-2014].

Paul Kabbash, William Buxton, and Abigail Sellen. Two-handed input in a compound task. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 417–423. ACM, 1994.

Dietrich Kammer, Jan Wojdziak, Mandy Keck, Rainer Groh, and Severin Taranko. Towards a formalization of multi-touch gestures. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 49–58. ACM, 2010.

Kenrick Kin, Björn Hartmann, and Maneesh Agrawala. Two-handed marking menus for multitouch devices. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(3):16, 2011a.

Kenrick Kin, Tom Miller, Björn Bollensdorff, Tony DeRose, Björn Hartmann, and Maneesh Agrawala. Eden: a professional multitouch tool for constructing virtual organic environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1343–1352. ACM, 2011b.

Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 258–264. ACM, 1994.

Anthony Martinet, Gery Casiez, and Laurent Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 115–118. IEEE, 2010.

Roberto Martínez Maldonado, Judy Kay, and Kalina Yacef. Collaborative concept mapping at the tabletop. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 207–210. ACM, 2010.

Nimish Mehta, Kenneth C Smith, and FE Holmes. Feature extraction as a tool for computer input. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, pages 818–820. IEEE, 1982.

Tobias Meyer and Dominik Schmidt. Idwristbands: Ir-based user identification on multi-touch surfaces. In *ACM international conference on interactive tabletops and surfaces*, pages 277–278. ACM, 2010.

Microsoft. Microsoft menus. `http://msdn.microsoft.com/en-us/library/dn742392.aspx`, 2014. [Online; accessed 05-December-2014].

Meredith Ringel Morris, AJ Bernheim Brush, and Brian R Meyers. A field study of knowledge workers? use of interactive horizontal displays. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 105–112. IEEE, 2008.

Christian Muller-Tomfelde, Anja Wessels, and Claudia Schremmer. Tilted tabletops: In between horizontal and vertical workspaces. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 49–56. IEEE, 2008.

Halla H Olafsdottir, Caroline Appert, et al. Multi-touch gestures for discrete and continuous control. In *International Working Conference on Advanced Visual Interfaces (AVI)*, 2014.

Patrick Paczkowski, Julie Dorsey, Holly Rushmeier, and Min H Kim. Paper3d: bringing casual 3d modeling to a multi-touch interface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 23–32. ACM, 2014.

Yvonne Rogers and Siân Lindley. Collaborating around vertical and horizontal large interactive displays: which way is best? *Interacting with Computers*, 16(6):1133–1152, 2004.

Yvonne Rogers, William Hazlewood, Eli Blevis, and Youn-Kyung Lim. Finger talk: collaborative decision-making using talk and fingertip interaction around a tabletop display. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 1271–1274. ACM, 2004.

Volker Roth, Philipp Schmidt, and Benjamin Güldenring. The ir ring: authenticating users' touches on a multi-touch display. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 259–262. ACM, 2010.

Kathy Ryall, Clifton Forlines, Chia Shen, and Meredith Ringel Morris. Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 284–293. ACM, 2004.

Dan Saffer. *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. O'Reilly Media, Inc., 2008. ISBN 0596518390, 9780596518394.

Stacey D Scott, Karen D Grant, and Regan L Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *ECSCW 2003*, pages 159–178. Springer, 2003.

Qian Sun, Juncong Lin, Chi-Wing Fu, Sawako Kaijima, and Ying He. A multi-touch interface for fast architectural sketching and massing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 247–256. ACM, 2013.

Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 91–94. ACM, 2010.

Anthony Tang, Melanie Tory, Barry Po, Petra Neumann, and Sheelagh Carpendale. Collaborative coupling over tabletop displays. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1181–1190. ACM, 2006.

Aaron Toney and Bruce H Thomas. Considering reach in tangible and table top design. In *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on*, pages 2–pp. IEEE, 2006.

R.J. Trudeau. *Introduction to Graph Theory*. Dover Books on Mathematics. Dover Publications, 2013. ISBN 9780486318660. URL `http://books.google.de/books?id=eRLEAgAAQBAJ`.

Julie Wagner, Stéphane Huot, and Wendy Mackay. Bitouch and bipad: designing bimanual interaction for hand-held tablets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2317–2326. ACM, 2012.

Malte Weiss, Simon Voelker, Christine Sutter, and Jan Borchers. Benddesk: Dragging across the curve. In *ITS '10: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 1–10, New York, NY, USA, November 2010. ACM. ISBN 978-1-4503-0399-6. doi: 10.1145/1936652.1936654.

Raphael Wimmer, Florian Schulz, Fabian Hennecke, Sebastian Boring, and Heinrich Hußmann. Curve: Blending horizontal and vertical interactive surfaces. In *Adjunct Proceedings of the 4th IEEE Workshop on Tabletops and Interactive Surfaces (IEEE Tabletop 2009)*, 2009.

Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 193–202. ACM, 2003.

Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines, and Ravin Balakrishnan. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on*, pages 8–pp. IEEE, 2006.

Wei Shin Yu, Hiske van Duinen, and Simon C Gandevia. Limits to the control of the human thumb and fingers in flexion and extension. *Journal of neurophysiology*, 103(1): 278–289, 2010.

Vladimir M Zatsiorsky, Zong-Ming Li, and Mark L Latash. Enslaving effects in multi-finger force production. *Experimental Brain Research*, 131(2):187–195, 2000.

Hong Zhang, Xing-Dong Yang, Barrett Ens, Hai-Ning Liang, Pierre Boulanger, and Pourang Irani. See me, see you: a lightweight method for discriminating user touches on tabletop displays. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2327–2336. ACM, 2012.

Typeset March 30, 2015