Integrating 2D Mouse Emulation with 3D Manipulation for Visualizations on a Multi-Touch Table

Luc Vlaming

November 23, 2010

university of groningen

Master's thesis
Institute of Mathematics & Computer Science
University of Groningen

Supervisors:
Dr. Tobias Isenberg (University of Groningen)
Dr. Sheelagh Carpendale (University of Calgary)

Copyright © 2010, Luc Vlaming
All rights reserved

Abstract

In this thesis I present an interaction technique that augments existing mouse-based visualization applications by enabling touch interaction. Previously, this transition to a multi-touch environment was difficult because the existing mouse emulation for touch surfaces was often insufficient to provide full functionality. My solution to this problem uses on-screen virtual mice, designed not only to provide mouse capabilities but also to act as toolglasses, taking advantage of their presence on the display itself. Developed together with a touch-enabled 3D window management system, my approach permits touch interaction with both the 3D windowing environment as well as with the contents of the individual windows contained therein. I describe the implementation of my technique that augments the VisLink 3D visualization environment to demonstrate how to enable multi-touch capabilities on all visualizations written with the popular prefuse visualization toolkit.

Parts of this thesis have been published in Vlaming et al. [2010].

Acknowledgements

First and foremost, I would like to thank my supervisors, Sheelagh Carpendale at the Interactions Lab of the University of Calgary, where this work was done, and Tobias Isenberg at my home university of Groningen. Without their excellent guidance and advice, this thesis would not have been possible.

I also want to thank Mark Hancock and Christopher Collins, from whom I used the frameworks for respectively the multi touch and VisLink software, and who helped me way too many times to count on all aspects of my research and writing up this story.

Furthermore, I am grateful to everyone else in the Interactions Lab, with special regard and many thanks to Miguel, June, Marian and John, whom gave comments and provided valuable help many times on my thesis. Many other iLabbers contributed valuable suggestion to my design and gave helpful input in testing the application. They also provided very valuable help in writing up my thesis. Generally everyone was very helpful and provided a very stimulating environment which became my home and "family" during my stay of six months, with everyone joking that the iLab family was working again on days like Family Day or Labour Day.

Last but not least, I would like to thank my friends and family at home whom provided very valuable help and support in writing my thesis, specifically Gieneke, my dad and mom, and Gjalt who spent countless hours helping me to get this finished.

Contents

Chapter 1

Introduction

With the sheer amounts of data that are available at the moment, it is difficult to sift through that data easily. This situation results in the need for people to get a grasp of the content of that data within a reasonable amount of time. Information visualization is used for this problem, which is defined as: "the use of computer-supported, interactive, visual representation of abstract data to amplify cognition [Card et al., 1999]." These visual representations, known as visualizations, exist in many different types and provide the ability to highlight different aspects of a dataset. Most visualizations are still used on desktop computers since they are designed to be used with a single keyboard and mouse. There are good reasons to use them on a desktop computer since one can provide, for example, precise pointing input and text input. There are, however, also downsides: for example, discussing a dataset with several colleagues using one or more visualizations becomes a problem when sitting around one screen and sharing mice and keyboard. With multi-touch (MT) interaction becoming ubiquitous, some of the problems with working with visualizations in a traditional computer setup can be improved upon. One advantage could be that direct-touch interaction could be used to control the visualizations instead of having to share one or more mice.

In this thesis, I examine how to enable MT interaction for several information visualization windows in a 3D virtual world. Since developing MT interaction techniques for information visualization is a really broad research area, I chose to focus my research to one particular well-respected visualization application: VisLink [Collins and Carpendale, 2007]. In this way, I can simultaneously limit the scope of the problem I am tackling and be sure the solutions I come up with have a practical application.

The interaction design developed in this thesis introduces a virtual mouse that works in tandem with MT interaction techniques to enable people to work with several visualization panels in a 3D virtual world on a MT screen. The design:

- allows 3D manipulation of visualization panels in the 3D virtual world,

- generalizes the support of 2D operations within each visualization panel such that any mouse activated visualization can be controlled,
- enables precise selection and manipulation within the panels,
- supports visualization comparisons, and
- allows for simultaneous operations on separate panels.

I start with the motivation behind this work and then give a brief overview of VisLink, the visualization system that I used as a basis based my work. Then I provide the design challenges that directed the design process after which I provide a brief overview of how the challenges in this work are approached.

1.1 Motivation

Currently, there are many visualizations available which can have more potential, for example, improved interaction speeds, when they are interacted with via direct-touch interaction on a touch table [Kin et al., 2009]. There is also potential in a collaborative setting because with direct-touch interaction with several people, the interactions are more fluid and interference is quickly resolved [Hornecker et al., 2008]. However, enabling MT interaction with visualizations is a very broad field and could have several solutions. Nevertheless, I proceeded because there are also several clear gains with enabling MT for the VisLink application, such as the ability to directly work with visualizations and to be able to quickly explore visualizations. In a collaborative setting, the gain of using MT interaction is the ability to be able to work with several visualizations at once, with several people.

1.2 VisLink

My work is situated in VisLink [Collins and Carpendale, 2007], an information visualization system that supports multiple visualizations built with the Prefuse toolkit [Heer et al., 2005]. These visualizations are placed in a 3D virtual world and can be moved around with special widgets for constrained movement of the visualizations in combination with keystroke combinations. The idea of VisLink is to allow a data analyst to explore and analyze relations between several visualizations. These visualizations can be used to show different aspects of one or more datasets. The relations between neighboring visualizations are indicated with edges drawn between the items, that are activated in the trigger visualization to the aspects in the neighboring visualization that relate. When an edge is created, it triggers through a “spreading activation” the activation of nodes on neighboring visualizations. This

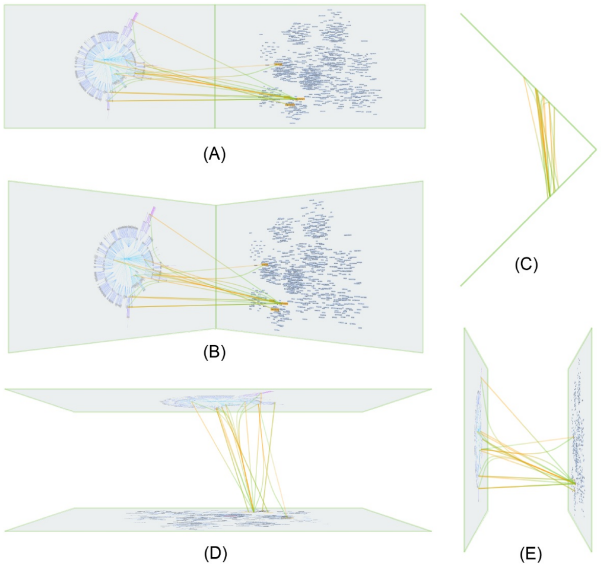


Figure 1.1: The different default views that are available in VisLink, for easing navigation in the 3D space. The views are (A) flat, (B) book, (C) book top, (D) top, and (E) side (image from [Collins and Carpendale, 2007]).

process continues until there is no activation left. In VisLink, five jumps after the initial activation are allowed before the propagation of activation stops. After the activation of a node, a forest of edges between visualizations is shown in which patterns could be recognized, allowing exploration of relationships between visualizations.

To allow movement of the visualizations in the 3D virtual world, widgets and default views are available. The widgets that are available are:

- Book-view opening: the visualization is grabbed on the left edge and rotated around the right edge, rotating a visualization like opening a page of a book;
- Garage-view opening: the visualization is grabbed on the bottom edge and rotated around top edge, rotating a visualization like opening a garage door; and
- Translation: a widget that appears when hovering over the center of the visualization. The visualization can then be translated along its normal vector with a special keystroke.

There are also some default views available (see Figure 1.1) to ease navigation in 3D space, which move the panels to a default configuration in 3D space. The default views that are available are:

- Flat view (A): the visualizations are adjacent to each other;
- Book view (B): the visualizations are oriented like the sides of a book;

- Book top view (C): the book view, but viewed from the top;
- Top view (D): the visualizations are parallel and viewed from the top; and
- Side view (E): the visualizations are parallel and viewed from the side.

To retrofit VisLink to work with MT input, several things need to be achieved. First, people need to be able to grab the visualization panels. Then, they need to be able to move the panels in the 3D virtual world, after which the core functionality, input to the visualization itself via MT input, needs to be created. There is a contradiction when naïvely enabling both manipulation of the panels and the input to the visualizations, since the techniques that exist to enable these requirements overlap in the input they use. Therefore, I created a new interaction design that allows both requirements to work in tandem.

1.3 Design

There is a delicate balance in designing a MT interaction technique with visualizations in a 3D virtual world. This is caused by the need to both provide input to the visualizations (2D input) and to manipulate the visualization panels in 3D in the 3D virtual world. The guidelines that directed the design process for my interaction technique are listed below:

Focus on visualization interaction. Although there should be a good balance between the 2D input and 3D manipulation, the main focus in this design should be on 2D input to the visualizations. This focus is required, since it provides the main functionality for the data analyst to do their work.

Support 2D operations that are visualization-indifferent. Input to the visualizations is the focus of the interaction. Every visualization, however, has its own requirements in terms of functionality it provides (panning, zooming, etc). Mapping this functionality directly to a MT interaction technique requires a taxonomy of the functionalities all visualizations generally provide. This is rather difficult as can be seen, for example, in [Yi et al., 2007]. While most visualizations share a common set of functionalities, each also provides a set of very specific or unique interactions, making the required number of functionalities that need to be mapped large. Therefore my design should provide a way to provide input to the visualizations that is visualization-indifferent. I chose to provide this by emulating mouse input, preventing changes to the visualizations. This also prevents the requirement to learn gestures to access the functionalities provided by the visualizations.

Allow reflexive use of the 2D input technique. When people are experts in using the mouse emulation technique, it would be beneficial if they could use the technique reflexively. This could enable someone to work faster with the technique by removing overhead of having to look at the area of interaction.

Enable precise selection and manipulation in 2D. While the use of MT interaction promises a very direct connection between people and the visualizations, the use of hands and fingers to provide the MT input can also cause a loss of precision [Voida et al., 2009]. That is, when touching a MT screen with our fingers, they usually touch far more than one pixel, thereby making it ambiguous which pixel location should be used as touch location. Moreover, by touching somewhere on the screen one also covers the area where input is provided, thus making it problematic to do corrections to provide the intended location.

Since most visualizations expect precise mouse input, my design should provide precise input to the visualizations. Also, my design should provide precise input to the visualization without occluding the area that is actually being interacted with.

Allow for simultaneous input when possible. In the information visualization field collaboration already occurs frequently. Until relatively recently, people have had to rely on physical prints of information for collaboration, and now mostly use programs like Microsoft PowerPoint and Excel. Recently also several systems have been created that support computer aided collaborative analysis, for example: Cambiera [Isenberg and Fisher, 2009], CoCoNutTrix [Isenberg et al., 2009], and Lark [Tobiasz et al., 2009]. Also, a study by Isenberg et al. [2008] suggests that parallel work increases perceived efficiency. Moreover, the study states that people have a strong tendency to work in parallel, and dynamically switch between individual and joint work when necessary. This illustrates that computer aided collaboration is possible and can be beneficial to the information analysis process. Therefore, I think it can be beneficial to allow collaboration in my interaction design. While collaborative interaction is not the main priority of this thesis, supporting some form of collaboration can improve the interaction design.

All visualizations supported by VisLink have already been implemented to use a single mouse. Moreover, they can be interacted with individually by placing them in a 3D virtual world. Therefore collaboration can be supported by my design without drastic changes to the visualizations. Thus, my design should support multiple people using several visualizations.

Support visualization comparisons. One of the potential benefits of having the visualization panels in a 3D virtual world is that panels can be placed side-by-side or grouped in a way that makes comparison possible. With the interplanar edges

used by VisLink, the spatial placement of visualizations becomes more important, since the interpretation of the interplanar edges can reveal new relations in the dataset. Therefore, spatial comparisons between several visualizations in the 3D virtual world should be supported.

Allow full 3D manipulation of visualization panels. In VisLink, precise mouse input combined with keyboard strokes are required to interact with the program, to manipulate the visualizations in 3D in the 3D virtual world. However, with a MT setup this becomes a problem since no keyboard is available and touch input is generally not as precise as mouse input [Voida et al., 2009]. Therefore, my design should support a way to manipulate the visualization panels in 3D in the 3D virtual world via MT input, to still allow someone to see a visualization panel from any view required, and to be able to make the spatial comparisons required by the previous characteristic.

1.4 Approach

There are three main challenges that need to be addressed in order to allow someone to work with VisLink via MT: interaction with the 3D virtual world, interaction with the visualization itself, and precise input for the visualization.

For the first problem, interaction with the 3D virtual world, first the visualization panel is given some thickness and several buttons for functionality which was partially also provided by VisLink. To then enable 3D manipulation in the 3D virtual world, I use the Sticky Tools technique Hancock et al. [2009]. This technique, however, proved to be too unconstrained. Moreover, it did not allow precise movement of a visualization panel. Neither did it allow precise placement of several visualization panels in the virtual world, which is required to do spatial comparisons between visualizations. Therefore, I created several layouts, in which only constrained movement along a specified path in the virtual world is allowed (see Figure 1.2 for an example).

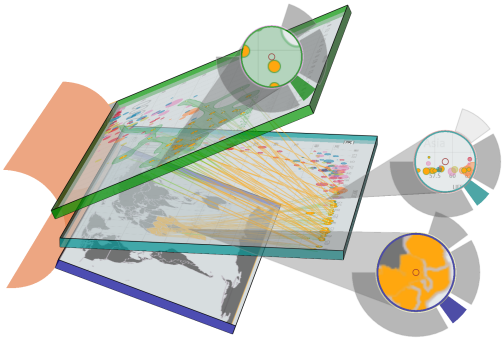


Figure 1.2: A restricted layout in which the panels can only move along a path.

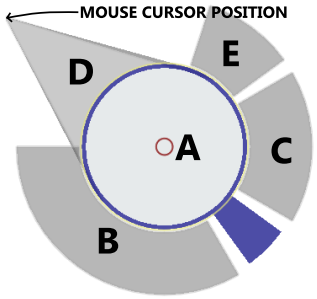


Figure 1.3: The design of the virtual mouse tool: it has a base (A), two buttons (B and C), a cone pointing towards the actual mouse cursor (D) and a visualization specific button to zoom (E).

For the second problem, enabling input for the visualizations, I propose a technique which emulates mouse input for visualizations on a MT touch screen via a virtual tool. This tool resides in screen space; it has a base (region A, see Figure 1.3) which can be "held" with a finger, two buttons (regions B and C), a cone (region D) pointing towards the actual cursor position on the visualization, and a visualization specific button to control zooming (region E). The advantage of using a virtual-tool in screen-space is that it can provide relative input, similar to a normal mouse. With the relative input the tool can enable precise input, addressing the third problem. Moreover, because it is a tool, one virtual mouse can be provided per visualization, allowing concurrent input to several visualizations.

Figure 1.4 shows all solutions incorporated into a program, in a typical screenshot of the application running.

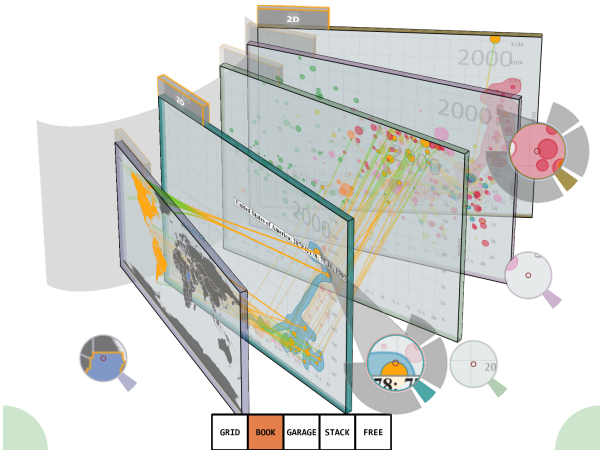


Figure 1.4: A screenshot of the program created in this project. It shows a typical scenario with multiple visualizations.

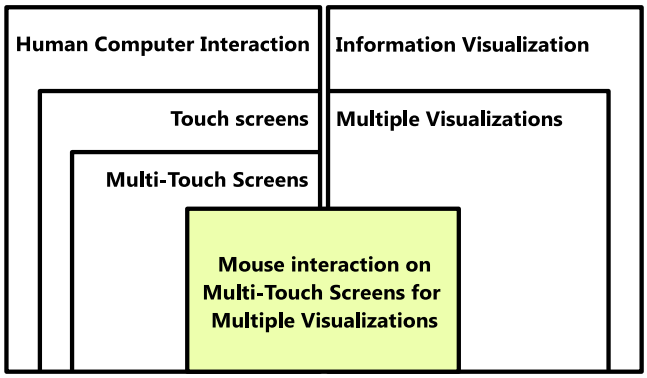


Figure 1.5: The scope and context of this research. It falls in both the Human-Computer Interaction and Information Visualization fields.

This work contributes to the field of Human-Computer Interaction (HCI) by designing a tool that provides a link between humans and computers. It also contributes to HCI by designing an interaction technique that allows several people to explore visualizations on a MT table in a new way. Furthermore, this work contributes to the field of Information Visualization (InfoVis) by designing the interaction technique to work with visualizations (see Figure 1.5).

1.5 Organization

The remainder of this thesis is organized as follows. First, Chapter 2 presents the related work on touch screens and the interaction techniques that are developed for it. In addition, the related work on window management systems, mouse emulation, precision enhancement, and information visualization is provided. Then, Chapter 3 presents a design for a touch-usable version of a visualization panel plus a solution for manipulation of the visualizations in the 3D virtual world. After that, the design of the virtual mouse is presented in Chapter 4. Then, a short discussion on the implementation, a scenario and an evaluation is provided in Chapter 5. Lastly, Chapter 6 presents my contributions, conclusions, and future work.

Chapter 2

Related Work

This chapter discusses how my work relates to research in the field. In discussing these related works I emphasize their influence on my research.

In order to develop a MT interaction technique, a MT capable screen is needed. Therefore, Section 2.1 reviews the different technologies that enable MT input with respect to the requirements of this project. After that, Section 2.2 provides an overview of the MT manipulation techniques that currently exist that support MT interaction with objects in the 3D virtual world. Then, Section 2.3 provides an overview of the window management techniques that exist; the techniques that allow input to 2D windows in a 3D virtual world. Thereafter, Section 2.4 provides an overview of the mouse emulation techniques that exist for MT input. Then, Section 2.5 provides an overview of the techniques available to enhance precision with MT input. Lastly, Section 2.6 provides an overview of collaborative information visualization techniques related to MT interaction.

2.1 Technologies

In accordance with the guidelines in the introduction, there are two main characteristics a MT screen should provide: many touches concurrently on the screen, and collaboration with several people using the screen concurrently with several touches each. Many touches must be supported to allow someone to use MT manipulation techniques. Collaboration is a specific requirement, because while several touches in total may be provided, there may not be support for enough touches per person to use MT manipulation techniques concurrently.

There are several different main forms of technologies that provide touch input. These technologies can be divided into groups according to the methods they use. Some are based on optical methods, which use imaging to provide touch input. Others use resistive methods, where two thin conductive layers are used to provide touch input.

A third group is capacitive methods, that use the electrostatic field of a human to determine the location of touch. Each of these groups will be discussed in the next sections, looking at whether or not they provide the characteristics required.

2.1.1 Optical

Within the group of optical based methods, there are various ways to register a touch. Most of the methods discussed here use infrared light (IR) to enable touch detection, together with a camera which is sensitive to IR. However, the way touch is registered is different for each. Also, the types of interactions that are possible differ. For example, some allow the person to also use tagged objects or pens, while others only allow finger interaction.

With all these techniques, cameras are used in combination with computer vision algorithms to register touches. The cameras that are used differ per technique, however, across all techniques most of them could be using the same cameras. With most of the techniques the number of touch points is only limited by the ability of the sensor to distinguish them.

To display graphics on the touch surface, most of the optical methods use a projector. With all methods except Digital Vision Touch, the projector is beneath the screen, and a projection surface is put on top of the acrylic. Digital Vision Touch uses a frame around a normal display to make it touch sensitive.

Diffuse Illumination

Diffuse Illumination (DI) is a technique where IR is used to make everything near the touch surface visible to the sensor. However, when the distance from the touch surface gets bigger, it is more difficult to see the object. Interaction takes place on an acrylic plate surrounded by a box wherein all the other elements, like the camera and projector, are placed. The system is a closed box, light can only get in and out through the supporting acrylic plate (see Figure 2.1).

With DI the hand is made visible directly by either making the hand very bright by shining IR on it from the inside with a light source, or very dark, by having the background very bright and the hand cast a shadow. Then, via computer vision software, the actual fingertips can be recognized as touches on the surface. When an IR source is used inside the box, a diffuser is used to even the spread of the source.

The DI technique has both characteristics needed: it supports many touches and allows collaboration. Since touches are visible directly, as many touches as can be distinguished can be used concurrently. For collaboration, both a horizontal and vertical setup is possible. Also, the size of the display can be any size, enabling this to be used as, for example, a table around which people can gather to collaborate.

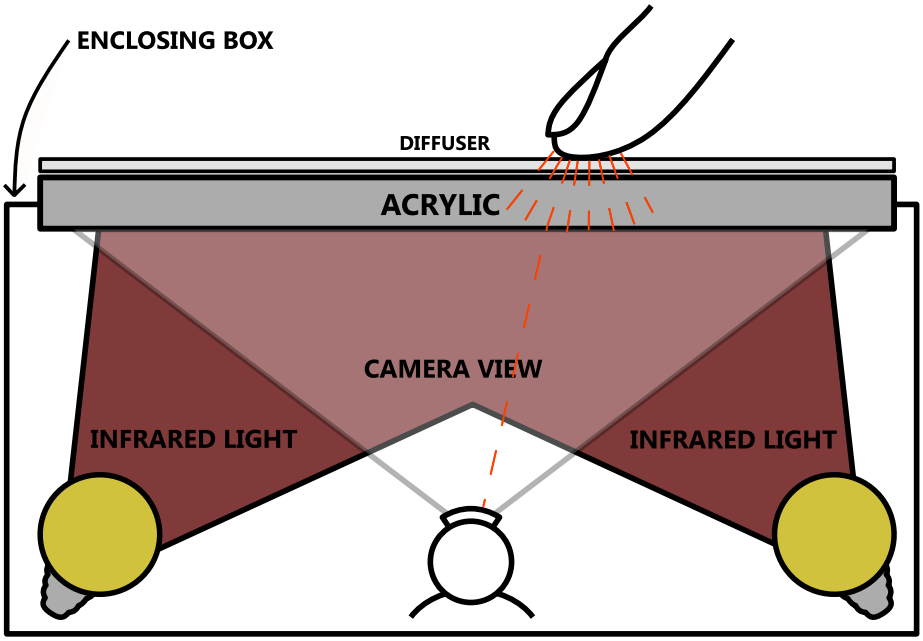


Figure 2.1: A possible setup for DI. An acrylic plate with a diffuser is used as a touch surface. When people touch down on the acrylic, the IR is reflected down towards the camera, which can then be recognized as a touch.

However, there cannot be distinguished between a finger or an elbow touching the surface, obstructing things like leaning on the touch surface.

One extra advantage of this technique is that the person can use not only their fingers to interact with the screen, but also objects with fiducial tags on them [Bencina and Kaltenbrunner, 2005]. Because reflection is used, no actual pressure is required for a touch or object to be seen by the computer vision software, so objects with a fiducial placed on the surface can also be recognized by recognizing the pattern of the fiducial. This is beneficial in my case, because this allows objects with fiducials on them to be used for, for example, mouse input.

Another advantage is that the position of the whole hand can be detected, so information about which touch points belong to which hand can be available. This provides valuable information in a collaboration setting, allowing the program to distinguish, for example, if there is a conflict with two people trying to interact with the same object on the screen.

Frustrated Total Internal Reflection

Frustrated Total Internal Reflection (FTIR) is a technique where IR is reflected down toward the camera as soon as a person touches the surface [Han, 2005]. Interaction

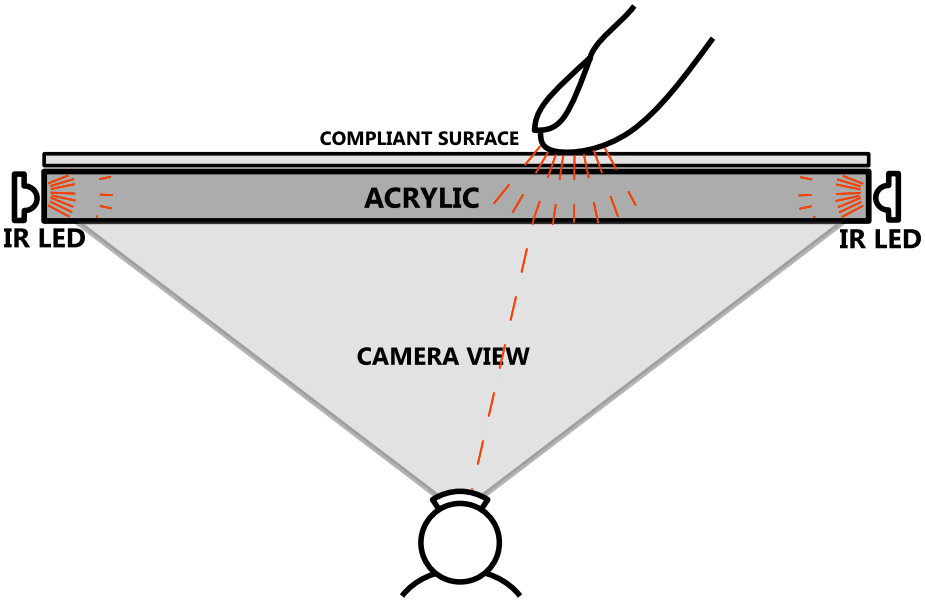


Figure 2.2: The schematic for the FTIR technique; an acrylic plate with a compliant surface is used as a touch surface. When the person touches the screen, the light is no longer reflected inside the acrylic, but reflected down towards the camera.

takes place on an acrylic plate, with a camera beneath the acrylic, and IR LEDs aside the acrylic. FTIR uses a phenomenon called total internal reflection to keep the light inside the acrylic when it is not touched. This reflection occurs when a ray of light hits the border of a transparent medium at an angle larger than the "critical" angle (with respect to the normal of the surface). The critical angle is the angle at which no light is sent out of the surface anymore, but all light is reflected internally. The critical angle is determined by the material's refraction index. The phenomenon only occurs when light is traveling from a material with a high refraction index to one with a low refraction index (for example from glass to air).

The FTIR technique requires a person to press their finger on the acrylic, changing the refraction index and thus the critical angle. This effect causes most light to be sent out of the surface toward the camera (see Figure 2.2), enabling touch recognition by looking at the bright spots of the sensor image. To increase the amount of light that is sent out by a touch, a "compliant surface" is used to increase the connectivity between the finger and the glass. A compliant surface is a material that changes the refraction index when the surface is touched, and lets the light through when the surface is not touched.

The FTIR technique provides both characteristics needed (many touches, collaboration), and allows for the same use as a DI setup: the setup can be any size and can be placed horizontal or vertical. However, objects with fiducial tags on them cannot be

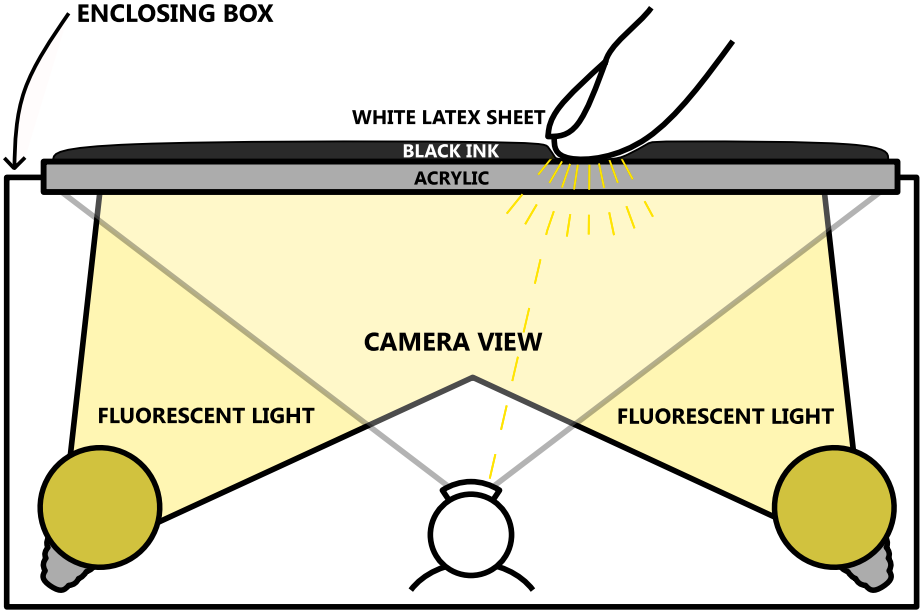


Figure 2.3: The schematic for the LDS technique; an acrylic plate with a contained liquid on top of it is used as a touch surface. The liquid is contained by a white latex sheet, which reflects the fluorescent light.

used, preventing, for example, the use of objects for mouse input.

One extra advantage of a FTIR setup is that it can provide pressure information of each touch on the surface. When a finger presses down harder, the contact between the acrylic and the finger is increased, thereby sending out more light to the sensor which can be translated to pressure information.

Liquid Displacement Sensing

Liquid Displacement Sensing (LDS) is a technique where an acrylic plate with a pressurized pouch on top is used as a touch surface, in combination with fluorescent light from below [Hilliges et al., 2008] (see Figure 2.3). The liquid in the pouch is black ink and the top of the pouch is a white latex sheet. When the liquid is pushed away, the latex sheet reflects the fluorescent light to the sensor.

The LDS technique provides the many touches characteristic like DI and FTIR by using a camera. However, there may be problems with the setup and size for collaboration use. The technique can only be used horizontally and needs regular maintenance to keep the pouch in good condition. Moreover, big sizes may not be possible due to the pouch: when the size gets bigger, it becomes more problematic to keep the pouch under pressure, and ripples will also start to appear when the pouch is touched. Last, objects with fiducial tags on them can not be used, preventing, for

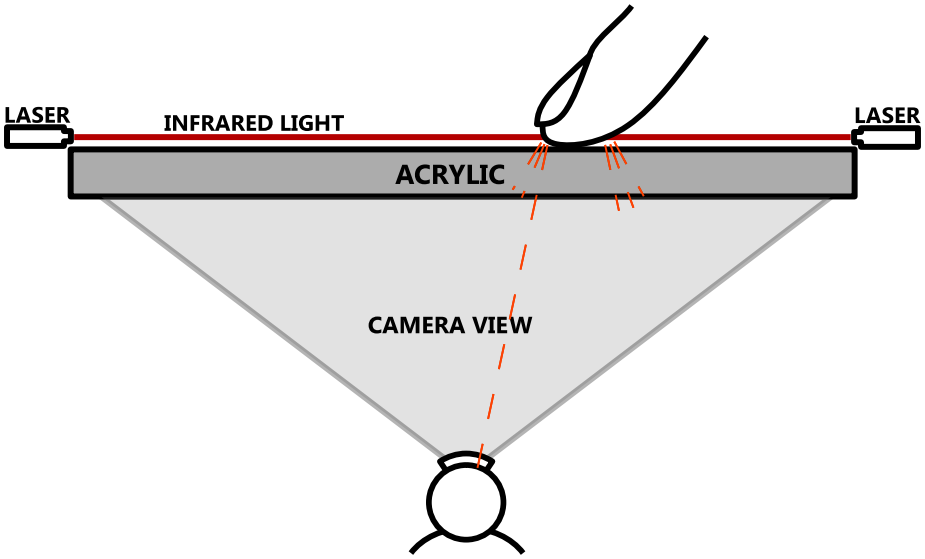


Figure 2.4: The LLP technique. Lasers are placed in the corners of the display just above the touch-surface. When the person touches the surface, the laser light is reflected down towards the camera.

example, the use of objects for mouse input.

An advantage LDS has over, for example, DI is that it can provide an indication of pressure. When more liquid is pushed away, the amount of light that is sent to the sensor on that spot will be higher, providing pressure information.

Laser Light Plane

Laser Light Plane (LLP) is a technique where lasers are placed in the corners of a display, just above the display screen itself (0.1mm). Special types of lasers are used to effectively flood the screen with laser light (see Figure 2.4). When a person touches their finger on the screen (or just above), the laser field gets interrupted and reflects down toward the camera. This effect makes the contours of each touch visible.

However, problems can occur with "shadows," creating ambiguities. When a person touches somewhere on the screen, it creates a shadow from every laser behind the finger. Shadowing can be countered by placing more infrared lasers on the sides of the screen (that is why normally four lasers are used, one in each corner), but if used by multiple people at the same time, shadowing can still cause problems.

The LLP technique provides the collaboration characteristic partly, but can have problems with many touches depending on how many lasers are used. Collaboration can be a problem with these screens due to the limited amount of touches that are supported. Also, objects with fiducial tags on them cannot be used because only the

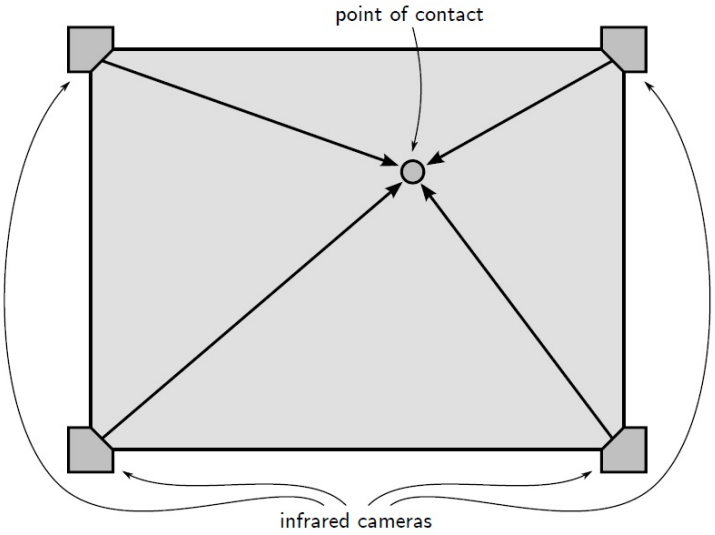


Figure 2.5: The DViT technique. It uses cameras in each corner of the display, together with infrared light strips on all sides of the screen. When a person touches the screen, it looks at the shadows created by the touch to determine the touch position (image from [ten Cate, 2009]).

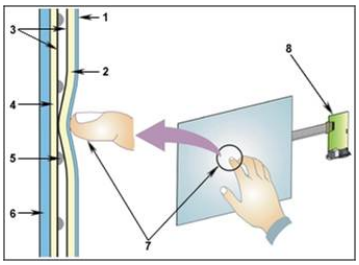
contours of each object are visible, preventing, for example, the use of objects for mouse input.

Digital Vision Touch

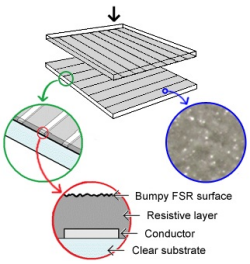
Digital Vision Touch (DViT) is a technique where cameras are placed in each corner of the display to triangulate the touch position [Smart Technologies, 2003]. Infrared light strips are placed along all sides of the displays. The position of a touch is determined by looking at the shadow a touch creates (for each camera) and thereby triangulating the position of the touch on the screen (see Figure 2.5). The whole setup is placed in a frame, and can be placed around a regular screen.

The DViT technique provides neither of the characteristics required unfortunately, since for collaboration several touches are required and also the many touches characteristic is not supported. Because only four cameras are used, only a maximum of two touches can reliably be detected, thereby disabling true collaboration, or real MT input.

The advantage of this system is the ease at which a standard screen is transformed into a touch screen, provided that the screen is a form factor the frames are produced in. Also, the touch screens are relatively easy to transport, and can be used at any angle. Big displays are no problem either, so some form of collaboration could be possible, as long as people are carefully taking turns interacting with the system.



(a) The analog resistive technique; it has two separate transparent conductive layers, which connect when a person presses down on the screen.



(b) The IFSR technique. Rather than reading the values of a touch at the border, wires are placed in a grid over the display, in two layers.

Figure 2.6: The resistive and IFSR techniques (images from [Rosenberg and Perlin, 2009]).

2.1.2 Resistive

Resistive touch technology has been around since at least 1999. Until recently, only analog resistive touch technology was available [Hong et al., 1999]. This technology works by having two transparent layers of conductive material with a transparent gradient resistor in between. The gradient resistor is placed between the two layers in a regular grid. Transparent materials are used, so the technology can be integrated into the actual screen.

When a person touches the screen, it closes the gap between the two conductive layers (see Figure 2.6), and from the resulting connection the location can be calculated. The position can be calculated by looking at the value the touch "generates" at each side of both of the two layers. However, this technique allows only one touch, since only one value can be read from each side of a layer, thereby not providing any of the required characteristics.

Recently, Rosenberg and Perlin [2009] introduced the Interpolating Force-Sensitive Resistance (IFSR) technique. This technique uses the same idea as the analog resistive touch technique; however, rather than only having wires at the borders for sensing, each layer has wires in one direction, for example bottom layer horizontal and top layer vertical (see Figure 2.6). Also a bumpy force-sensitive resistive material is used rather than the gradient resistor. This enables pressure sensing and makes it a pressure display rather than a sensor for one touch. It is a pressure sensor because, for each crossing of wires, a pressure value can now be determined. Recognizing touch with this technique can then be done in the same ways as with the optical based methods.

The IFSR technique provides both characteristics (many touches and collaboration), provided that the display can be created in large enough sizes. While this should be possible, currently no size over 22 inch is available, making the available

displays rather small to collaborate on. The screens can be placed and moved anywhere easily, and can be used horizontally and vertically, making them easy to be used in a collaborative setting.

An advantage this technique has over most optical methods, is that it supports a stylus and can also differentiate between stylus and finger input by looking at the size of the pressure area.

2.1.3 Capacitive

Within the group of capacitive based methods, there are two main techniques: Projected Capacitance and DiamondTouch. To make a display, the DiamondTouch uses top projection while Projected Capacitance uses an integrated display.

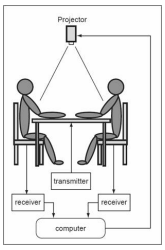
Projected Capacitance

Projected Capacitance (PC) is a technique where an etched grid is used to form a grid pattern of electrodes. Voltage is applied to create a grid of capacitors. When a person brings their finger close to the grid, the local electrostatic field changes, which can be measured. By measuring this field for each capacitor, the touch location can be determined accurately. Because a grid is used and several distortions can be measured at once, this enables MT interaction.

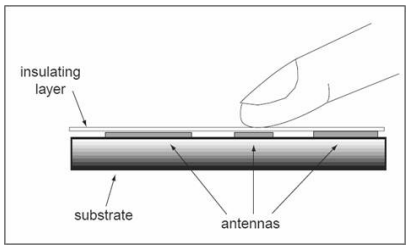
The PC technique has both characteristics needed: it support many touches, and in theory allows collaboration. However, because the expense to make these displays in large sizes is very high, these displays are unfortunately not available at sizes large enough to collaborate around. Also, the technique can only recognize fingers or special conductive objects as touches, disabling use of tagged objects to provide, for example, mouse input.

DiamondTouch

DiamondTouch (DT) is a technique where an small current in an electrical circuit is used to identify touches. A small current flows from the table to a chairpad through the person. The table contains a regular grid of two layers of antennas (one horizontal and one vertical). When someone touches the table, a (better) connection is created, allowing more current to flow back to the table again which can be measured at the antennas that are touched (see Figure 2.7). The system uses a specific frequency for each person, thereby enabling unique identification of the touches of each person. Because separate horizontal and vertical antennas are used, when a user touches with two fingers, only a bounding box of these touches can be determined. However, with some adaptations to the algorithm, it can be deduced in which corners of the bounding box the touches are [Bérard and Laurillau, 2009].



(a) The complete setup. A small amount of current is flowing from the pad the person sits on, through the person, to the table. When the person touches the table, a loop is created.



(b) The table setup. A regular grid of antennas is used to register touch. When the person touches the table, current flows through a horizontal and vertical antenna.

Figure 2.7: The DiamondTouch technique (images from [Dietz and Leigh, 2001]).

The DT technique provides the collaboration characteristic; it can even uniquely identify which person is touching. However, a maximum of two touches per person is allowed.

An extra advantage this system has, is that only touches from people which are using the table (sitting on a pad) are actually registered, so objects that are placed on the table, or touches by other people will not disrupt the interaction.

2.1.4 Summary

Summarizing all the techniques to register touch input, only a few are practical to use. The Projected Capacitance and Interpolating Force-Sensitive Resistance techniques are dropped from consideration for this thesis because they cannot provide big enough sizes to be able to collaborate. The Projected Capacitance method is too expensive to use at the required sizes, and displays with the Interpolating Force-Sensitive Resistance technique are not produced at the required sizes yet. The DiamondTouch technique is dropped because only two touches can be used per person, and MT manipulation needs to be possible which would probably require more than two touches per person. Therefore, also the Digital Vision Touch technique is dropped since this technique only supports two touches in total. The Liquid Displacement Sensing technique is not used either since it is not yet commercially available. The choice of technique to use then is between Laser Light Plane, Frustrated Total Internal Reflection, and Diffuse Illumination. In the lab I worked in there were displays with the Frustrated Total Internal Reflection technique (a SmartTable) and Diffuse Illumination (a Microsoft Surface) technique available. In the end, both of these devices are supported by my technique, but the Microsoft Surface is used to make the videos because this device displays colors better thereby making it better for visualizations to be shown on.

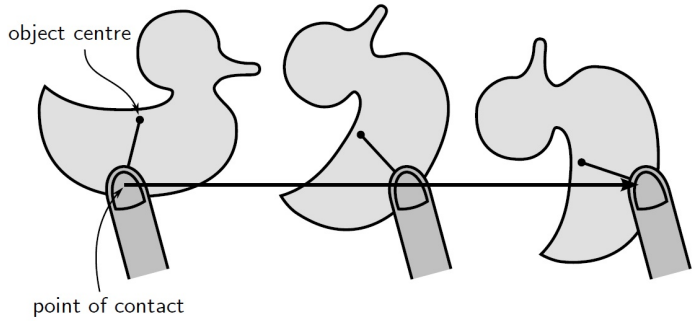


Figure 2.8: The RNT technique. The object behaves as if it has a center of mass, allowing someone to rotate and translate with a single finger (image from [ten Cate, 2009]).

2.2 Multi-Touch Manipulation

For MT displays, many interaction techniques have been developed. In accordance with the guidelines from the introduction, the main characteristics that these techniques should provide is 3D manipulation of objects in the 3D virtual world and the ability to concurrently use the technique on a MT screen with several people.

In this section, interaction techniques that provide manipulation in 3D will be reviewed. Since some of these techniques are based on 2D manipulations techniques, those will be covered first.

2.2.1 2D Manipulation

To provide 2D manipulation of objects there are a few widely used solutions to interact with objects: RNT, TM, and RST.

Rotate n' Translate A common one-touch interaction technique is rotate n' translate (RNT) [Kruger et al., 2005], a technique for rotating and translating an object at the same time with one touch point (see Figure 2.8). RNT emulates an object with a center of mass, allowing for, for example, pushing or rotating of an object with one touch point.

Traditional Moded Second, there is the Traditional Moded (TM) technique, which does either rotation or translation. The mode is determined through dedicated regions on the object itself. The corners provide rotation around the center of the object, while grabbing anywhere else does translation only.

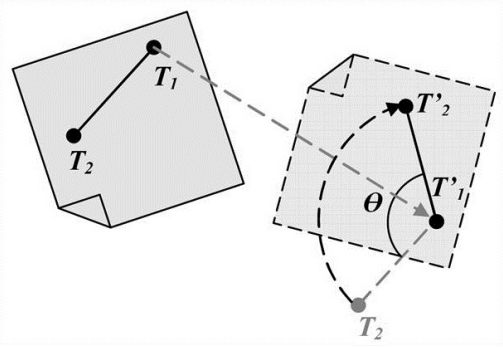


Figure 2.9: Schematic of the RST technique. The first touch controls the position, while the second controls rotation and scale of the object (image from [Hancock et al., 2007]).

Rotate-Scale-Translate There also is the Rotate-Scale-Translate (RST) technique [Hancock et al., 2006] which uses one or two touches to do provide rotation, scaling and translation. With one touch, the object is translated. With two touches, the first touch still does translation, while the second touch does scaling and rotation. The scale is determined by the distance between the two touches, while the rotation is determined by the angle the two touches make (see Figure 2.9).

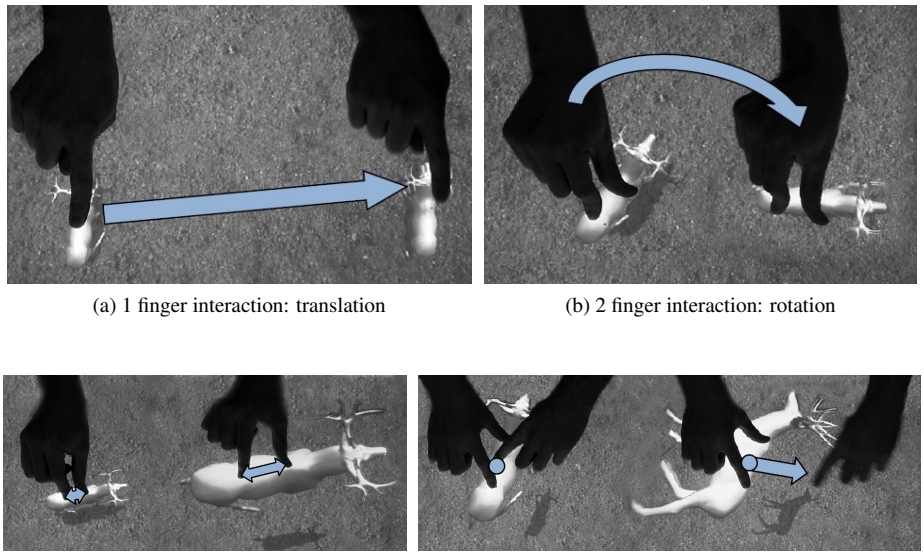
2.2.2 3D Manipulation

For 3D manipulation of objects with MT, there are several interaction techniques available. Most of them provide 3D manipulation, but not all provide control over all 6 degrees of freedom (6DOF) of movement.

Sticky Tools

Several techniques were developed by Hancock et al. [2007, 2009] which improve the RNT technique by extending it to 3D manipulation. Hancock et al. provide two techniques which use in total two or three touches to provide full control of the 6DOF of movement. Moreover, their studies show that the feeling of "picking up" an object is better approximated by using more fingers. Therefore, they designed the *Sticky Tools* technique which introduces the *sticky fingers* and *opposable thumbs* concepts to provide a technique that provides full control over 6DOF of movement with three touches in total [Hancock et al., 2009]. The sticky finger concept means that someone's fingers stays in touch with the virtual object in the location they initially contact.

With the Sticky Tools technique, the order in which someone touches down their fingers on a virtual object are used to assign specific control to each finger. The technique behaves as follows (see also Figure 2.10):



(a) 1 finger interaction: translation

(b) 2 finger interaction: rotation

(c) 2 finger interaction: lifting

(d) 3 finger interaction: rotation in 3D

Figure 2.10: The Sticky Tools interaction technique (images from [Hancock et al., 2009]).

- if an object is touched and dragged with one finger, 2D translation is performed.
- if an object is touched and dragged with two fingers, the object is translated in 3D (in screen-space) and rotated along one axis. The translation in z (lifting of the object) is determined by the distance between the fingers. The rotation is along the z-axis (with respect to the screen space), which is determined by the angle the first and second finger make.
- if an object is touched and dragged with three fingers, it performs translation in all directions, and rotation in all directions. The third finger provides the remaining two rotation options, by introducing a relative rotation around the x- and y-axis.

The Sticky Tools technique has both characteristics needed: it provides full control of the 6DOF of movement, and allows concurrent manipulation of several objects at once. Moreover, in the Sticky Tools technique also the *virtual tools* concept is introduced, which allows objects to function as tools and thereby influence each other. Using the virtual tools concept allows richer manipulations to take place like, for example, pushing an object. The virtual tools concept is very useful in my work, since this allows an object to be used to provide functionality with visual feedback. This is in contrast to other solutions where one would maybe not have visual cues on the existing functionality.

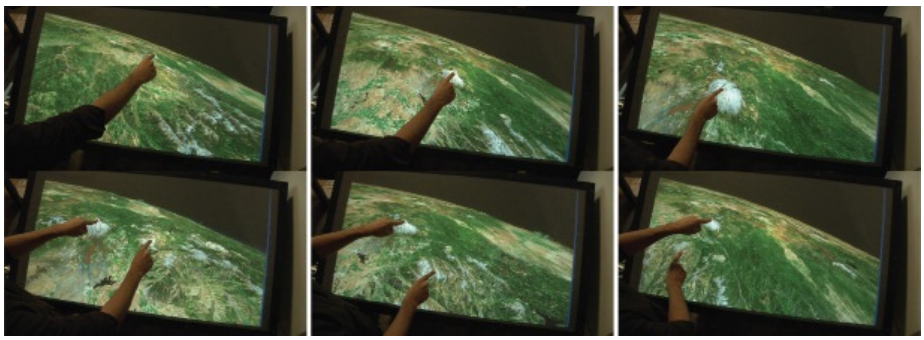


Figure 2.11: The screen-space direct manipulation interaction technique interaction in two scenarios for interacting with a globe. The top scenario moves the mountain closer with a one-touch movement, whereas the bottom scenario moves the mountain further away and rotates the earth around the mountain at the same time (image from [Reisman et al., 2009]).

2D and 3D Direct Manipulation

Reisman et al. [2009] introduce an alternative interaction technique, which extends the RST technique to provide full control over all 6DOF of movement. This technique allows people to directly manipulate objects with 6DOF by using 3 or more points, using an equation solver in combination with error minimization (see Figure 2.11). However, by using an equation solver, problems then can arise with interactions which have multiple solutions to solve the equation, for example, when an object is exhibiting rotational exhaustion. It is ambiguous, for example, when all touches are placed on one side of the plane, the object is already faced head on, and one finger is moved away. Therefore, constraints are introduced to limit these problems. One example of a constraint that can be used is with a one-touch interaction, then only translation in x and y is allowed.

This technique has both characteristics needed: it provides full control of the 6DOF of movement, and allows concurrent manipulation of several objects at once. Moreover, this technique provides direct control over the manipulation of all 6DOF of movement.

Physics For 3D Manipulation

An alternative strategy to implicitly enable 3D manipulation is introduced by Wilson et al. [2008] by using a physics engine to enable control over objects in the 3D virtual world. This technique represents each contact point on the surface as a physical beam that interacts with the virtual world (see Figure 2.12). The contours of each contact point are bundles touching down into the world, which exert a certain amount of force on the objects they are touching. This method enables someone, when they move their

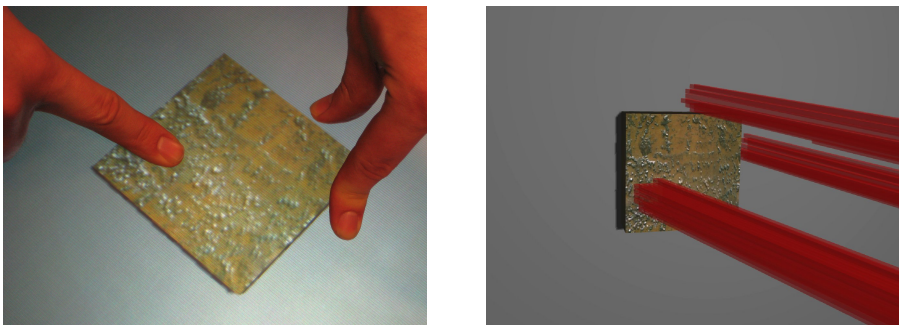


Figure 2.12: Using a physics engine for MT manipulation. The contours of each touch are represented as bundles touching down on the objects, in this way exerting force on the objects thereby enabling manipulation (image from [Wilson et al., 2008]).

fingers (the contact points), to interact with the virtual world, thus allowing that person for instance to move objects, rotate them, or stack them. Because physics is used, none of the interactions are hardcoded. However, a full physics engine is needed to allow someone to move the objects since gravity, friction, etc. all need to be available in order for this to work.

This technique has the characteristic of allowing concurrent manipulation of several objects at once. However, it does not provide full control over all 6DOF of movement, since an object cannot be lifted pushing on it.

There is another technique which also uses physics simulations to provide manipulation of all 6DOF of movement, by allowing someone to also interact above the table [Hilliges et al., 2009]. When someone makes a pinching gesture, the object is grabbed, after which is can be moved around to any position, while still interacting with other objects in the 3D virtual world (see Figure 2.13). The object is then lifted by lifting the hand.

Although the second technique does provide 6DOF movement, it requires addi-

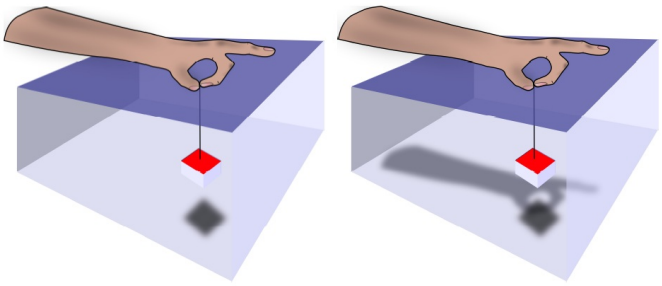


Figure 2.13: The hand of someone making a pinching gesture, thereby grabbing the object, and afterwards lifting it (image from [Hilliges et al., 2009]).

tional hardware to be able to see above the MT surface. Moreover, this technique does not yet provide very precise movement (in depth) as the sensors are not very precise in measuring depth.

2.2.3 Summary

Summarizing all 3D manipulation techniques, I decided that the Sticky Tools technique is the best candidate to be used. Because VisLink uses a 3D virtual world without a real top or bottom, it is impractical to use a physics technique since this would require defining a gravity vector. Moreover, using a physics technique also poses all other kinds of problems: are panels allowed to move through each other, are the interplanar edges also represented in the physics engine, etc . The screen-space formulation technique is impractical since there are ambiguities on how certain manipulations should be resolved and it is also rather difficult to implement. Therefore, the Sticky Tools technique is used to provide 3D manipulation of objects in the 3D virtual world.

2.3 Window Management Systems

In accordance with the guidelines from the introduction, mouse emulation needs to be provided in order to enable input to the visualizations themselves. Since the visualization panels exist in a 3D virtual world, this input has to happen in tandem with 3D manipulation in the 3D virtual world. Therefore, the current existing 3D window management systems are reviewed. 3D Window management systems provide both 3D manipulation and 2D mouse input via a 3D virtual world, making them useful examples of how 2D input within a 3D virtual world can be provided.

Miramar

Miramar is a 3D workplace where a small virtual world is provided in which a person can keep their documents and organize them [Light and Miller, 2002]. There is an animated transition between the 2D desktop and the 3D workspace to tie them together. Documents in the workspace can be organized in several ways, including relative and absolute placement. The placement of the documents is done with a 3D drag and drop operation, and one can move around in the environment with simple mouse and keyboard operations.

In other words, Miramar provides the ability to provide 2D input in a separate mode, while working with the 3D desktop requires a transition to the 3D workspace. So unfortunately this does not fit any of the requirements.

Figure 2.14: Project Looking Glass: a 3D window manager, allowing 3D placement of windows, writing notes on the back of windows, and 3D objects to be used in the window manager.

Project Looking Glass

Project Looking Glass is designed to create a 3D window manager [Sun, 2010]. It supports movement of windows to the side of the virtual world (the windows are rotated around the x-axis), allows writing notes on the back of windows by flipping them, and adds translucency to windows when they are not in use (see Figure 2.14). Moreover, when an application is adapted to the 3D window manager, the application can provide 3D objects to the window manager for the person to interact with. They present, for example, a CD browsing application which allows someone to flip through the pile in the 3D window manager itself.

In other words, Looking Glass provides the 2D input to applications by locating where is clicked. Mouse input is sent to applications as soon as an application is activated. The coordinates are then transformed to the 2D coordinates of the application, and sent through. The movement of application windows is achieved by using keystrokes with mouse input.

BumpTop

BumpTop is a 3D desktop manager [Agarawala and Balakrishnan, 2006] which allows someone to organize their computer desktop more like a real physical desk with four walls around it (see Figure 2.15). All documents on the desktop have physical properties so one can, for instance, “bump” documents on the desktop around, or

Figure 2.15: BumpTop: a 3D desktop manager, allowing people to organize their desktop like they would do in the real world (image from [Agarawala and Balakrishnan, 2006]).

documents can be pinned to a wall.

There are also some interaction techniques to organize your desktop better. For example, there is a gesture to create a pile from a set of objects, which can then be shown as a grid or be flipped through as a pile of cards. Objects can be resized to increase (or decrease) importance. Recently, MT support was also added to BumpTop.

However, interaction with applications is not possible. While the initial version of BumpTop allows "minimized" windows on your desktop, there is no reference on how this works or how to interact with them. As they say: "we focus on interaction with icons, and not on how windowed applications would live in our interface."

VisLink

Although VisLink was already discussed, here a short review on its window management technique is provided for completeness. VisLink is a system which has several visualization panels in a 3D virtual world [Collins and Carpendale, 2007]. These panels can be manipulated in 3D with widgets and the use of keyboard strokes (see Figure 2.16).

Input to the visualizations themselves is provided via the mouse. All mouse cursor activity is translated to 2D input by choosing the visualization that is beneath the mouse cursor, and then translating that input to 2D coordinates in the coordinate system of that visualization.

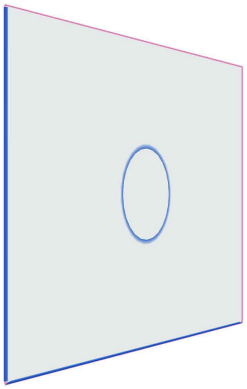


Figure 2.16: The VisLink widgets to move an object around. The circle provides translation along the normal of the panel, while the side widgets provide rotation along two axis.

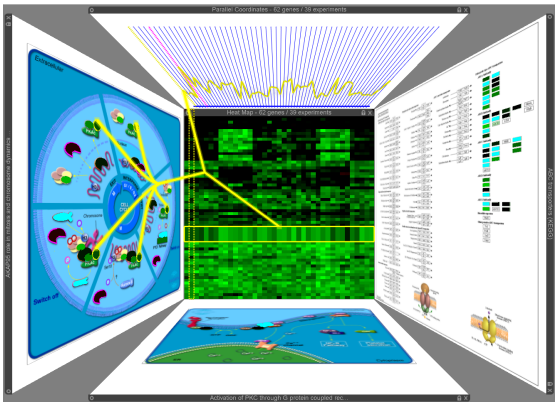


Figure 2.17: Caleydo: a visual analysis tool for gene expressions, here showing links between three "windows" for a particular query.

Caleydo

Caleydo is a tool for visual analysis of gene expression data [Lex et al., 2010]. The idea of Caleydo is based on The task Gallery, where a box is provided with the running windows on the sides [Robertson et al., 2000]. A small part of Caleydo uses a 3D view in which a box is provided in which several types of "windows" can be shown (see Figure 2.17). Using this box, on each side visualizations can be shown which show a particular view on a dataset. Then, by hovering or selection, queries to this system can be made to for example filter on this dataset.

While this is not a 3D window manager, it does provide input to visualizations in a 3D virtual world. Here all mouse activity is transformed to the coordinate system of the visualization the mouse is on-top of, and then sent to that visualization to be handled.

Summary

Summarizing all 3D window management techniques, only Project Looking Glass and VisLink really provide the combination of 2D input to applications while also providing 3D manipulation of the windows in the 3D virtual world. However, there is a key difference between the two systems though on how input is sent. With VisLink, no activation of a visualization panel is required in order for input to be sent to the visualization. With Looking Glass, an application first needs to be activated in order to receive input. One of the characteristics that should be supported is collaborative work, so it should be possible to use several visualizations at once. Therefore, in my design visualization panels must always be able to receive input without requiring activation.

2.4 Mouse Emulation

In accordance with the guidelines from the introduction, mouse emulation needs to be provided, and concurrent input to several visualizations needs to be possible. Since a MT interaction technique is designed, this mouse input needs to be provided via a MT display. Therefore, the mouse emulation techniques that work with MT are reviewed.

DTMouse

The DTMouse method uses a three finger method to provide (one-button) mouse input [Esenther and Ryall, 2006]. When two fingers are put on the screen (see Figure 2.18), the mouse cursor hovers over the screen. Tapping a third finger toggles the mouse button. Using one finger drags the mouse cursor. This technique provides the full behavior of a one-button mouse: moving, pointing (hovering) and clicking.

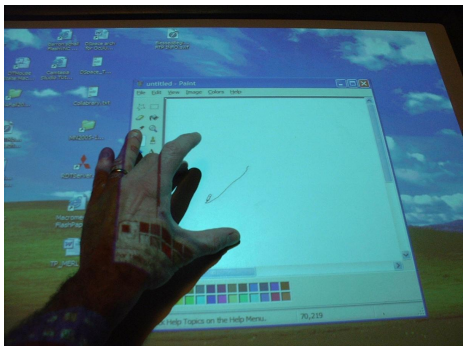


Figure 2.18: The DTMouse technique for mouse emulation. Here two fingers are used after using a third finger to toggle the left mouse button. The pointer is in the middle of the two touches (image from [Esenther and Ryall, 2006]).

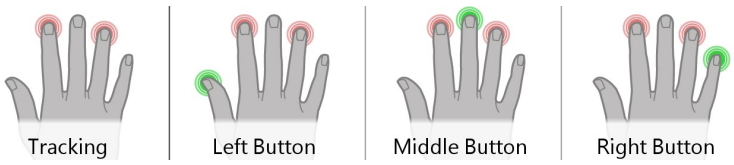


Figure 2.19: The Side Technique for mouse emulation. Which side of the two passive fingers is chosen determines which button is pressed (image from [Matejka et al., 2009]).

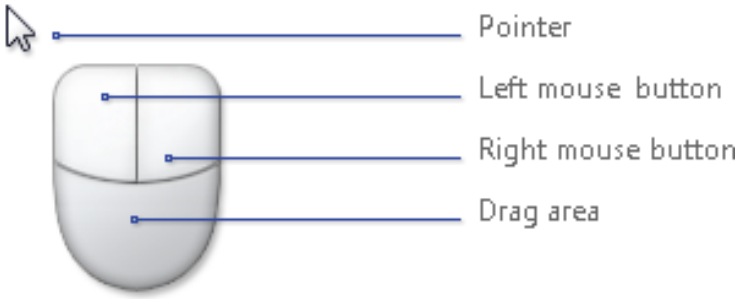


Figure 2.20: The Touch Pointer for mouse emulation. A two-button mouse tool is provided which can be dragged around to provide mouse input. (image from [Microsoft, 2010]).

Side Technique

Matejka et al. [2009] introduces a set of methods designed to emulate the mouse functionality of a three-button mouse using gestures. The distance between points determines if a left click, right click or middle click is performed (see Figure 2.19). One example of this behavior is the Side Technique, which uses the index and ring finger for tracking. The thumb activates the left button, the little finger activates the right button, and the middle finger activates the middle button.

Touch Pointer

Starting from Windows Vista, the Windows operating system also provides a tool that emulates mouse input via touch input. It is called the touch pointer [Microsoft, 2010], and is a tool that looks like a two-button mouse (see Figure 2.20). When the bottom is touched, the mouse cursor is dragged in a hover-state. When the left button is touched, the left mouse button click is emulated (and similarly for the right button).

Summary

Summarizing the mouse emulation techniques, they all provide some form of mouse emulation. However, the DTMouse only provides a one-button mouse, potentially limiting the practical use of existing visualizations by only providing one button. Moreover, both the DTMouse and the techniques by Matejka et al. allow only one emulated mouse to be used on a screen, disabling concurrent use of the system. The Touch Pointer also only provides one cursor, but there should be no problem to provide several when using it in a new system because it is designed as a virtual tool. For the design of my technique, the idea of using a virtual tool is good, because several tools then can provide concurrent input. However, the design of touch pointer is not practical: when several of the touch pointer mice would be used and people would cross each other, they would have to switch positions around the MT screen because their arms cross. This should be addressed somehow.

One advantage of the touch pointer is that all functionality can be deducted by looking at the tool, and visual cues are provided to show what is happening. With the DTMouse or the techniques by Matejka et al. this is not possible, requiring recall of the functionality that is available. Both techniques also require remembering what mode one is in, for example, if the left button is pressed. Therefore, my design should provide visual cues when required to prevent recall of the available functionality.

2.5 Precision Enhancement

In accordance with the guidelines from the introduction, in my design it should be possible to provide precise input to the visualizations. Moreover, the mouse cursor screen area should not be occluded when providing mouse input. Also, concurrent precise input to several visualizations should be possible. In this section several precise selection techniques are reviewed, after which in the summary the requirements will be discussed with respect to the techniques.

Early Dual Finger

Benko et al. [2006] introduce several techniques to provide precise touch input on MT devices:

- The Dual Finger Midpoint method puts the cursor halfway between two touch points, preventing occlusion and providing twice the precision of a single touch.
- The Dual Finger Stretch method creates a lens effect when the second finger touches down. The distance from the first to second finger controls the size of the lens, and the first finger provides the actual input point.

- Menu invoked precision enhancement methods are also provided, which use a menu (or hidden slider) to control the speed of the cursor.

Shift

Shift is a technique which, on ambiguous selection, displays a copy of the occluded part of the screen above the actual thumb, allowing the person to do corrections on its initial selection [Vogel and Baudisch, 2007].

Escape

This method uses gestures to select a single item in the vicinity of the touch point [Yatani et al., 2008]. After touchdown the person makes a small gesture in a certain direction, selecting the item they actually want to select. Visual cues are implemented to steer the gesture of the person. Several types of cues were explored, an arrow is used as visual cue in the final design, together with the highlighting of the current proposed selection.

To prevent the person from occluding the actual desired selection point, it is not necessary to touch on the objects. Touching close to it (just beneath it) works as well.

iLoupe

This technique features a lens which allows people to provide more precise input to the device [Voida et al., 2009]. This lens is displayed in a convenient area of the screen, or on a handheld device (iPodLoupe). The lens itself can be resized and moved, giving the ability to zoom into the content (or to show an overview of it) and to enable really precise input.

Summary

Summarizing these precision enhancement methods, all these methods provide precise input capabilities, however, only a subset of these also prevent occlusion, or allow collaboration. Of all techniques, the Dual Finger Midpoint, Shift, Escape, and iLoupe methods provide all requirements. However, none of these also take into account the problem that this should work in tandem with 3D manipulation of panels in the 3D virtual world. This creates problems when one of these techniques would be integrated with a 3D manipulation technique by the need to decide whether a touch somewhere is meant to manipulate an object in 3D or to provide input to it in 2D.

There are several good ideas in these techniques though: Shift provides a thumbnail to allow precise selection on ambiguous selection, and iLoupe offsets the input to a dedicated area to allow more precise input.

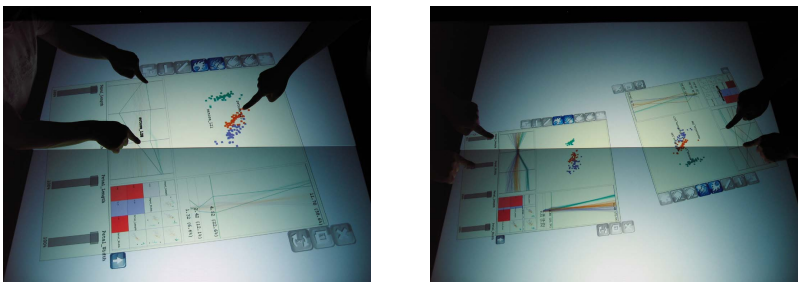


Figure 2.21: iPCA-CE: doing joint work on a dataset (left), and doing separate analysis of the same dataset (right) (images from [Jeong et al., 2009]).

2.6 Collaborative Information Visualization

In collaborative information visualization, lots of research is done. The research related to my work, is the work that combines collaborative information visualization with MT interaction, because my design should support simultaneous input on MT devices, which is the core part of what these systems provide.

In this section, several collaborative information visualization applications are reviewed in order to have some perspective on how well my interaction design supports collaborative information visualization.

iPCA-CE

iPCA-CE is a tool that supports interactive data analysis using principal components analysis (PCA) on a tabletop display [Jeong et al., 2009]. iPCA-CE supports both shared and separate workspaces, allowing people to do joint work and also work on their own (see Figure 2.21). Moreover, work with partitioned datasets is supported too, which allows people to focus on a specific dataset instead of the combined dataset.

Looking at concurrent input, iPCA-CE allows simultaneous input by providing each person with its own view on the dataset, or by allowing people to simultaneously interact with different UI parts of a single dataset.

Lark

Lark is a system which allows collaborative work on a dataset using a MT screen [Tobiasz et al., 2009]. Lark has a pipeline from which can be branched at all stages. There are four stages: the analytical abstraction, spatial layout, presentation and the view itself. For every stage, branches can be created and parameters for that stage can be set. Lark supports several collaboration styles: tightly and loosely coupled work, and parallel work. The collaboration styles are supported by allowing each view to be resized, moved and rotated to suit the particular style (see Figure 2.22). Also, by

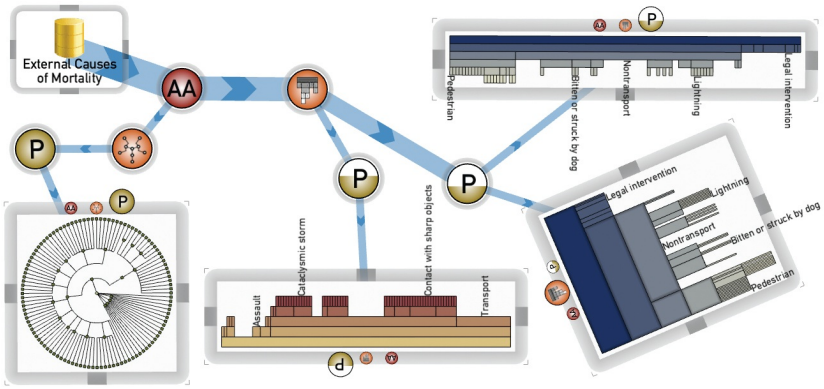


Figure 2.22: Larks collaborative visualization environment: single data set External Causes of Mortality, four views, plus the meta-visualization (image from [Tobiasz et al., 2009]).

choosing where the pipeline is branched one can scope the influence of their changes, only affecting certain views.

Looking at concurrent input, Lark supports simultaneous input by scoping interaction, having each person interact with one view. However, it is not stated if people can interact with one view concurrently, probably leaving conflict resolution to the analysts themselves.

CoCoNutTrix

CoCoNutTrix is a tool that retrofits the NodeTrix social-network-analysis tool to enable multi-user interaction in collaborative environments [Isenberg et al., 2009]. CoCoNutTrix allows movement of nodes, and grouping by doing a lasso figure around the group. Mice are used instead of a MT screens, to have a low-cost effective solution, and to minimize retrofitting effort.

While mice are used to provide input, CoCoNutTrix uses concurrent input to allow people to move several nodes at once, making it an interesting case to look at. Also, conflict resolution is explicitly left to the people using the system themselves. This was done to save time adapting the original system, but also because this is often a successful resolution method [Tse et al., 2004]. This is interesting because it can also save time adapting the VisLink system to allow concurrent input.

Cambiera

Cambiera is a tool that allows collaborative searching through many documents to do data analysis [Isenberg and Fisher, 2009]. Each person has its own search actions, which are shown as bars on the screen. These bars are extended with columns for

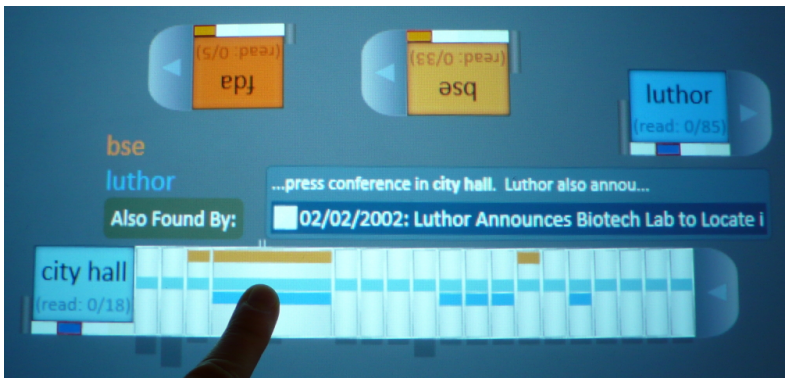


Figure 2.23: Cambiera: a tool that allows collaborative searching to do data analysis (image from [Isenberg and Fisher, 2009]).

each document that has been found, and rows in each column that show in which other searches the document has been found (see Figure 2.23). Also, for each author a different color is used. Documents can be dragged out of the search bar to become a floating representation of the document, which can be enlarged to view the document itself.

Looking at concurrent input, people can work on their own, while they get feedback of the actions someone else is doing. However, it is not stated if people can interact with one search bar concurrently, probably leaving conflict resolution to the analysts themselves.

Summary

Summarizing these collaboration systems, it is either not explicitly defined how conflict resolution takes place, or left to social protocols. In my interaction design, conflict resolution will be left to social protocols since conflict resolution is not my main focus.

Moreover, in some of the reviewed systems there are numerous methods to allow various styles of collaboration, from loosely coupled to parallel. To support this interaction, these systems are adapted extensively. Since the main focus of my thesis is not on supporting collaboration, but just enabling simultaneous input when possible, the various collaboration styles will not be supported explicitly.

Furthermore, in most systems there are also various ways to inform analysts of what someone else has done, or control the scope of items that is influenced by someone's action. While this is good to support collaborative work, this would also require extensive design to support and is therefore not supported explicitly.

2.7 Summary

Gathering from Section 2.1, both a Microsoft Surface and SmartTable will be used to test my interaction design. Then, the Sticky Tools technique will be used to provide manipulation of all 6DOF of movement in the 3D virtual world. To be able to provide concurrent 2D input to several visualizations at once, the VisLink concept will be used. This choice implies to send input to a visualization as soon as there is interaction in the screen region of that panel. Also, a tool should be used to provide input providing visual cues when necessary, to prevent recall of the available functionality. For precise input, there is no particular existing technique that can be used. However, there are several good ideas to incorporate in a new technique: using a thumbnail to allow precise selection, and offsetting input to allow for more precise input. The various techniques required to allow collaboration will not be supported explicitly; only concurrent input to several visualizations should be supported.

Chapter 3

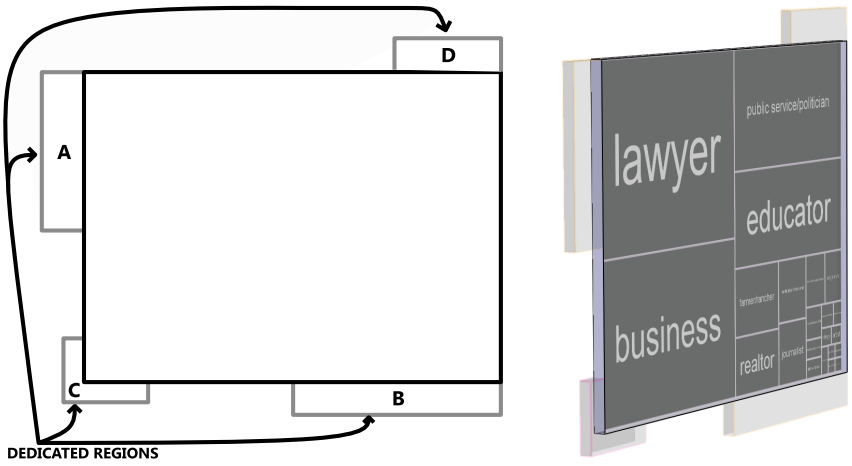
Interacting with visualization panels in 3D

As described earlier in the introduction, there are two conflicting sets of actions that need to be provided: 2D input to the visualizations and 3D manipulation of both the virtual camera and visualization panels. Although in the related work techniques are described to provide either, they cannot be used together since the input mappings overlap. Therefore, I created a design that provides both in harmony. In this chapter the 3D manipulation design is described.

In accordance with the guidelines from the introduction, my interaction design should support a way to manipulate visualization panels in 3D in the 3D virtual world via MT input. Also, there should be a focus on the visualization interaction. Moreover, my interaction design should also allow spatial comparisons between several visualizations. To create a good design for both the 2D input and the 3D manipulation, two extra guidelines are important:

Unhindered perception of visualizations. Visualizations are used in my interaction design and are its primary focus. Visualizations require, among other things, good color perception and high detail to be used effectively. Therefore, unhindered perception of the visualizations is important for the data analyst to correctly analyze the visualizations and be able to provide the 2D input required.

Use visual cues when required. Gathering from part of the summaries of Sections 2.2 and 2.4; virtual tools should be used to provide explicit awareness about functionality that is available. Furthermore, visual cues should be incorporated in the design when they have added value to prevent recall of available functionality and providing feedback on, for example, which mode someone is in.



(a) Schematic of the design

(b) Screenshot of the design

Figure 3.1: The design for a visualization panel. It is designed like a window frame (around the visualization program), with buttons added on three sides and a corner. Each button has a unique function. Region A provides a bookview rotation, region B provides a garage-door rotation, and region C provides reshaping (width and height are controlled independently). Region D switches to a 2D mode of that panel

To allow 3D manipulation via MT input, first the panel needs to be redesigned to allow "grabbing" of the panel and be able to provide the same functionality it provided in VisLink. This is not currently possible since elements on the panel are, for example, too small. The new design of the panel is described in Section 3.1.

Then, a 3D manipulation technique for MT input needs to be designed. In the related work the Sticky Tools technique was chosen to provide this. However, during my research, first other techniques have been tried. Therefore, all the attempts to provide 3D manipulation will be described in Section 3.2.

Moreover, also spatial comparisons between several visualizations should be possible, so Section 3.3 provides the design decisions to support these comparisons.

3.1 The Visualization Panel in 3D

Since MT is used, touch acquisition of the panel can be difficult when the panel is viewed at a steep angle and thus only a few pixels wide (or high) on the screen. Therefore, the panel is given a thickness (see Figure 3.1). To allow MT to be used to access the functionality VisLink provided through widgets, this functionality is made available by adding 3D buttons to the panels. To retain unhindered perception of the visualization, the buttons are added at the side of the visualization panel. To provide a focus on the interaction with the visualization, the buttons are mostly transparent when

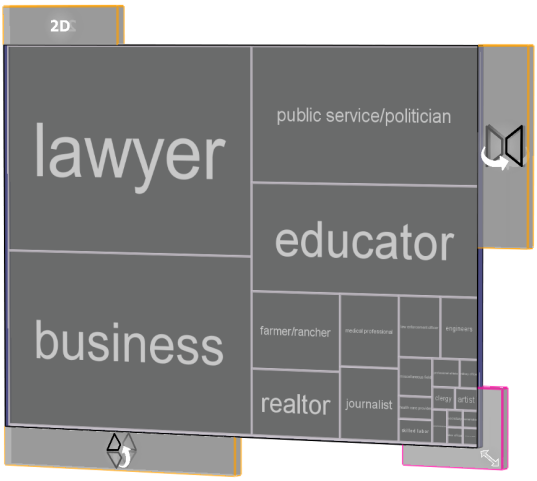


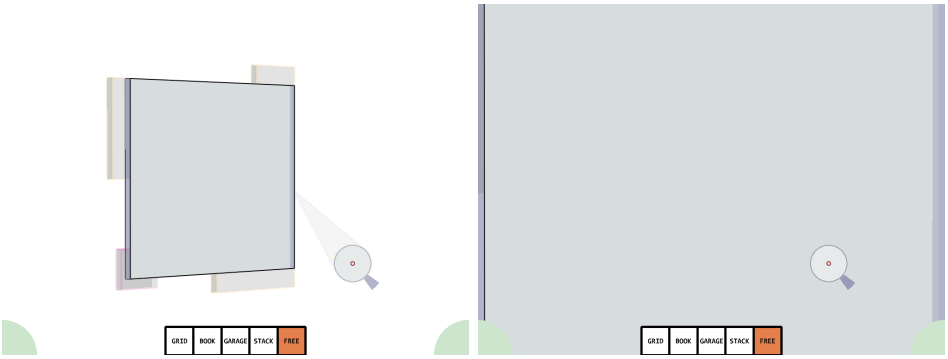
Figure 3.2: A screenshot of the design, when the panel is viewed from the back. The visualization is activated, so the frame has a unique color (blue). The buttons are more opaque and icons indicating their functionality are shown on the buttons.

the visualization is not in use.

When the visualization is activated (by someone interacting with it or via a tool indirectly interacting with it), the added elements become more opaque and have a lighter color (see Figure 3.2). Moreover, to provide a visual cue on what functionality is provided by each button, an icon will be shown on it when the panel is activated.

Furthermore, a visualization generally is directional (has an up direction and a reading direction for text). This causes problems when the panel is viewed from the back, because the reading direction is inverted. Therefore, the display of the visualization itself is flipped horizontally when the panel is viewed from the back, thus maintaining the correct reading direction on the screen (see Figure 3.2 compared to Figure 3.1b).

Another option that was considered to allow better access to the buttons, was putting the buttons parallel to the screen. This, however, creates the problem to define how to put the button parallel to the screen, since we get a conflict in the design. We want both to have accessible buttons, where we want to keep all elements at a "static" place to have consistency, and the added elements to not hinder the perception of the visualization. This causes problems when a user switches from viewing the front to the back of the panel. When this happens, the optimal direction to put the button parallel to the screen will flip. This will cause the button to be on the other side of the panel, making the region having a non-static orientation, generating undesirable behavior. Therefore, it is unfortunately not possible to provide buttons parallel to the screen, and instead the buttons are still provided at static positions around the panel (see Figure 3.1).



(a) The camera navigation tools are in the left and right bottom corner, but also background interaction can be used for navigation.

(b) Here the camera navigation tools are only usable, the camera navigation via background interaction is not available anymore.

Figure 3.3: Virtual tools for navigation of the virtual camera.

3.2 3D Manipulation

To provide 3D manipulation for the system, two interactions need to be designed: navigation and manipulation [Bowman et al., 2004]. The first section, navigation, moves the virtual camera around. The second part, manipulation, provides 3D manipulation of the objects in the 3D virtual world.

3.2.1 Navigation

To move the virtual camera, a virtual tool is provided, which can be found in two corners of the screen (the left and right, see Figure 3.3a). The idea of a trackball metaphor is used to rotate the camera around two axes (x and y) when the tool is interacted with a single finger. When two fingers are used, the person zooms in (or out). Later it was observed that people also wanted to use the background to do these interactions, so when the person touches the background, this is also considered as a touch on the virtual tool (see Figure 3.3a). The tool is still kept for when one would zoom in until a point where the whole screen would be filled with objects and only interaction on the tools is available (see Figure 3.3b). With the tool, which is always displayed and on top of everything, one can then zoom out again.

3.2.2 Manipulation

To interact with the panels in the virtual world there are several possibilities. One can either create an interaction set where only restricted movement is allowed, and then add more control when necessary. For example, using more fingers to interact with an object removes movement restrictions. The other method is to use a free object

manipulation method (one that provides control over all 6DOF of movement at once) and then restrict movement when applicable.

During the creation of my interaction design, I first tried to create a restricted interaction. After that, I created an interaction design that uses the Sticky Tools technique, and restricted movement where applicable.

Restricted Interaction Set

During the development of the 3D manipulation interaction set, first several interaction sets were created which allowed only constrained movement of the panels along a path. The first design only allowed movement along a line, creating a stack. Only rotation and movement along the panels normal are allowed, plus scaling. With one touch, the panel is moved along its normal, thereby changing the order the panels are connected in and thus changing the interplanar edges shown between the visualizations. With two touches the visualization is rotated and scaled in place.

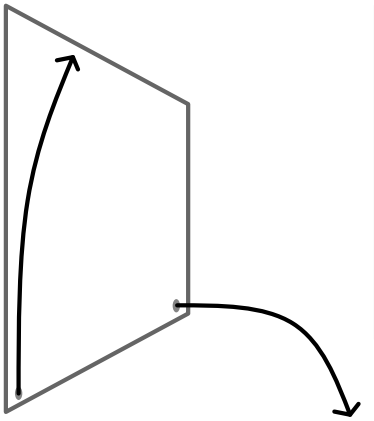
However, the rotations of panels VisLink provided are not available in this design. The rotation do, however, provide analytical power when creating for example a book-opening. Therefore, two methods were tried to provide these rotations.

First, I tried to create an algorithm that matches either a book-opening or garagedoor-opening depending on how two touch points on the object are dragged (see Figure 3.4), while the points are kept sticky in accordance to the *sticky fingers* concept [Hancock et al., 2009]. The problem here is to determine when to rotate around the x-axis, and when to rotate around the y-axis (see Figure 3.4). Some cases are clear (for example Figure 3.4b), whereas others would require scaling to have the interaction sticky (see Figure 3.4c). Moreover, some cases are difficult to determine what the operation would do at all (see Figures 3.4a and 3.4d).

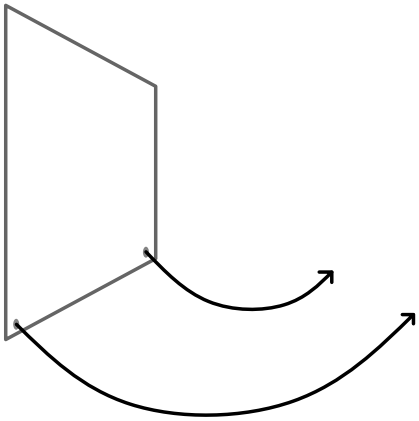
Free Interaction Set

In accordance with the related work, I chose the Sticky Tools technique to provide an integrated manipulation of all 6DOF of movement. However, when testing this technique to move panels around, it was quickly discovered that while this provides unconstrained movement, it also introduces severe problems with precisely positioning the panels. It has, for example, been shown that precisely aligning two objects in 2D, while having control over both translation and rotation, is difficult [Nacenta et al., 2009]. When having control over both translation and rotation in the 3D virtual world, precisely aligning an object will be even more difficult due to, for example, a skewed view on the object.

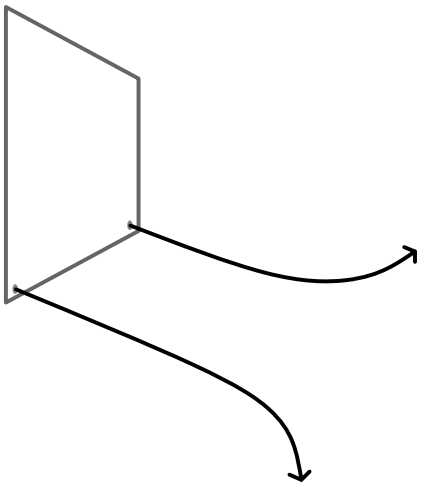
Therefore, the buttons from the restricted movement technique are provided here too, but now provide the specific original VisLink rotations (see Figure 3.1). Region A provides a bookview rotation; a rotation around the y-axis, around the right side



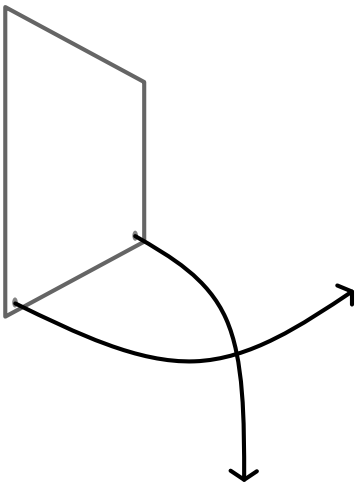
(a) Undefined operation, could be rotation around the z-axis of the object.



(b) A book opening rotation (rotation around y-axis of the object). The garage-door opening rotation (rotation around the x-axis of the object) is similar, but the points would be placed vertically instead of horizontally.

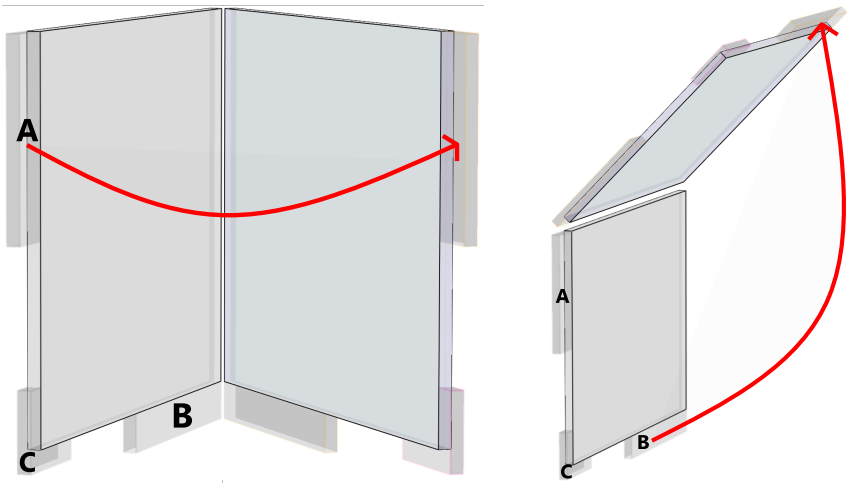


(c) A scaling operation, no significant rotation. To have the operation sticky (the touch points on the object stay relatively at the same position from where they started), the object needs to be scaled.



(d) Undefined operation, could be flipping the object horizontally, or rotating it 180 degrees.

Figure 3.4: Different operations when a two-finger rotation would be implemented (simplified representation of panel).



(a) The bookview rotation. The panel is rotated around the right side.

(b) The garage-door rotation. The panel is rotated around the top side.

Figure 3.5: The rotations the dedicated regions provide.

of the panel (see Figure 3.5a). Region B provides a garagedoor rotation; a rotation around the x-axis, around the top side of the panel (see Figure 3.5b). Moreover, a third button was added, region C, which provides reshaping of the panel, in the sense that width and height are controlled separately. This is of more value for visualizations than providing, for example, in-place rotation and scaling because for visualizations that divide the available space, like a sizemorph treemap, having a different aspect ratio for a visualization can be of importance (see Figure 3.6).

It was also observed that people associated multiple fingers interaction to try to resize the visualization panels. However, this was not limited to two fingers, also more than two fingers were used. This inspired the idea to use an averaging method to allow people to use as many fingers as preferred (two or more) to reshape a visualization panel, and offset it. The average of the fingers is taken and the average distance to the center is taken, controlling respectively the offset and the shape of the object. This is similar to what can be achieved with the method by Reisman et al. [2009], but here it is implemented directly instead of implicitly.

However, in the tests that followed it was determined that the Sticky Tools technique with the extra buttons still provided too much freedom. Moreover, even with the added buttons, it also took a lot of time to put the visualization panels in a configuration that was meaningful for visualization exploration. Therefore, in Section 3.3 my design for supporting spatial comparisons is explained which does not have these problems.

Figure 3.6: Examples of why reshaping of a visualization can be of importance. Visualizations can place their content differently based on the aspect ratio of the window they have. With this sizemorph treemap layout, the layout of the nodes is presented differently based on the aspect-ratio.

3.3 Spatial Comparisons

As explained in the previous section, the Sticky Tools technique does not support spatial comparisons very well. While working with the Sticky Tools technique and trying to do spatial comparisons, it also seemed beneficial to have several layouts of how panels are positioned in the 3D virtual world rather than only having one layout to work with. Having several layouts would allow someone to quickly switch between, for example, an overview of the visualizations and a detailed view on two of the visualizations looking at the interplanar edges between them.

To allow someone to quickly switch between two layouts, a simple toolbar is provided on the bottom of the screen with buttons to switch between the layouts (see Figure 3.7). All manipulations of the panels in a layout are saved, to allow quick switching between two layouts that have been setup to provide certain views on the visualizations.

To allow easy spatial comparisons, several constrained layouts (canonical views) were created, allowing movement only along a certain path, thereby keeping the panels

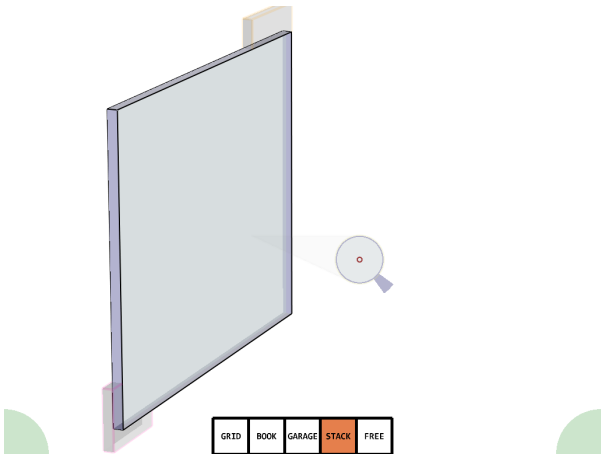
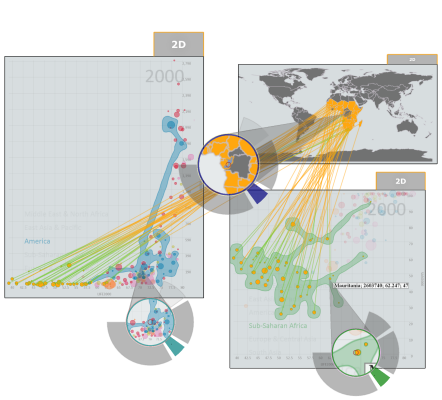
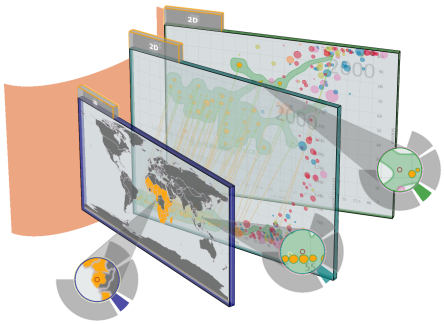


Figure 3.7: The toolbar (on the bottom) to quickly switch between different layouts.

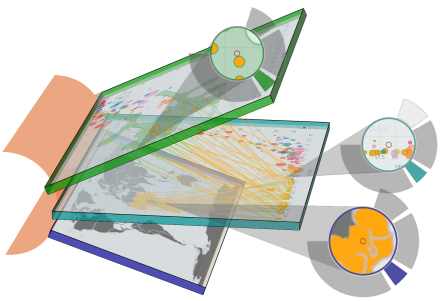


(a) The grid layout.

(b) The stack layout.



(c) The book view layout.



(d) The garage view layout.

Figure 3.8: The layouts that are available.

aligned. However, the "free" layout also remains available to allow an unconstrained arrangement when the provided constrained layouts would not be adequate. The layouts provided in this design were derived from the set of default views available in VisLink, but more were added. The added layouts were derived from working with a set of paper on a desk; this can, for example, be stacked or layed out adjacent in a grid. The constrained layouts available in my design are (see Figure 3.8):

- *book view*: the panels follow a bent line in the *x*-*z* panel; for example a half circle;
- *garage door*: the panels follow a bent line in the *x*-*y* panel; for example a half circle;
- *a grid*: all panels are in a 2D mode next to each other and can be moved on a panel; and
- *a stack*: all panels are on a straight line and can be moved along that line.

Other types of layouts were considered, but were not implemented:

- *peeling pages*: flipping through a set of pages in a book, in this way looking at a certain part of these pages (depending on where one grabs the pages); and
- *layered presentation*: visualizations are made transparent, thus several visualizations can be overlayed. Can be useful when the visualizations have the same spatial layout.

The manipulations that are available in these constrained layouts are:

- *move*: move one page to another place along the path;
- *reorder*: reorder the pages along the path;
- *zoom*: zoom in on a particular page;
- *control path*: change the path to be more bent;
- *offset*: make a page stick out; and
- *reshaping*: independently changing the width and height of a visualization.

Looking at the stack, bookview, and garagedoor layouts, the interactions are realized in the following way: moving and reordering of the panel is realized by interacting with the panel with a single touch and then moving the object along the path in the layout. The control of the path itself is realized by a virtual tool (see Figure 3.9), which can be interacted with. With the book and garage-door, using one touch on the virtual tool makes the path bend more.

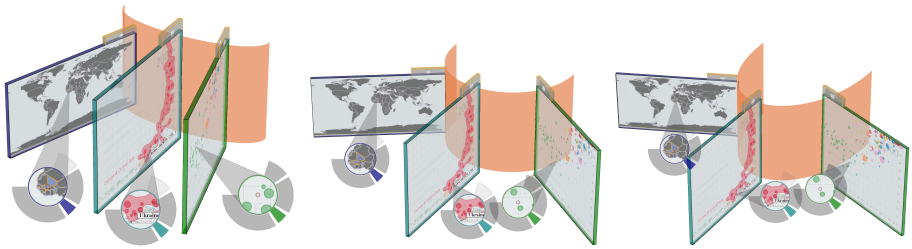


Figure 3.9: Screenshots of the virtual tool visualizing the path. The path along which the panels move can controlled by dragging with a single touch. From left to right the path is made more curved by dragging the tool forward.

For the grid configuration, the interactions are different. This is because there now is a 2D path "plane" rather than a single 3D curved path. The panels can be moved on the virtual plane they are on with a single touch. Furthermore, navigation of the virtual camera is adjusted here, because only a head-on view is allowed. When one touch is used, the whole scene is panned. When two touches are used, the zoom level changes.

In using the system, most people also requested a 2D mode to provide a traditional view on a single visualization. At first this was provided by a double-tap on a visualization. However, since this prove to be very difficult to do with MT, an extra button on the top was added instead (region D, see Figure 3.1). Tapping this button goes to 2D mode of that visualization, tapping the button again (which is also shown in 2D mode) switches back to the old layout.

3.4 Summary

In this chapter, a new design for the visualization panels is introduced. This allows grabbing of the panel from any side, and provides several buttons to provide functionality to rotate, reshape and switch between 2D mode and 3D world. Moreover, the Sticky Tools technique is used to provide 3D manipulation of all 6DOF of movement. Furthermore, to provide an effective way to do spatial comparisons between visualizations, several layouts are introduced which provide constrained movement of panels. These layouts allow the panel to only move along a predefined path. This path can be controlled in some of the layouts to be more bent.

Chapter 4

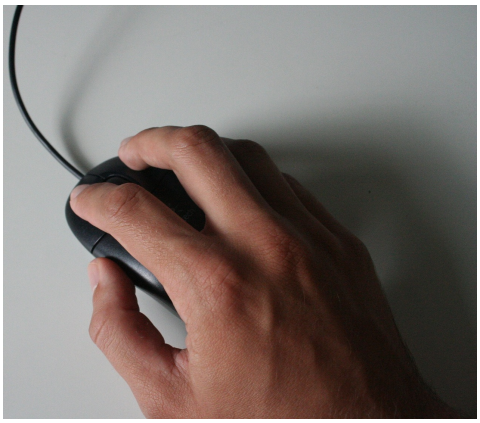
Interacting Within Visualization Panels in 2D

In addition to the 3D interaction techniques, interaction with the 2D visualizations that are displayed within the panels also needs to be possible. In accordance with the guidelines from Chapters 1 and 3, it should be possible to focus on the visualization, to interact with the visualization, and to have unhindered perception of the visualization. Furthermore, visual cues should be incorporated in the design when they have added value. Moreover, mouse input should be provided via mouse emulation, precise mouse input should be possible, and simultaneous input should be possible. Lastly, it is preferred to be able to provide mouse-emulation input reflexively as soon as someone is an expert in using the technique.

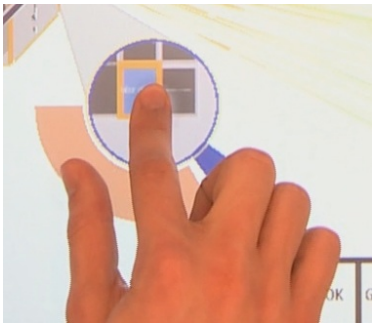
In order to provide the 2D functionality of a physical mouse (see Figure 4.1a-b) with a MT technique, it is informative to review the characteristics of a mouse. It has a base on which the palm of the hand rests, allowing someone to move the mouse. It has two buttons and a scroll wheel which can be pressed independently. In addition, movement of the mouse is detected by a high-precision sensor. The types of input these characteristics provide are scrolling and relative pointing in several modes. These modes are controlled by the button states and are generally time-dependent, allowing for differentiation between click, double-click, drag, and hover.

To provide mouse emulation for visualizations in the 3D virtual world, I chose to design a virtual tool (see Figure 4.1c). This has several benefits: it can easily provide visual elements that show which functionalities are available, it can provide relative input by having part of the tool point to the actual cursor location, and can provide scaled input to enhance precision. Also, by using a virtual tool to provide mouse-input emulation, there is the advantage that one virtual tool per visualization can be provided. This enables the ability to provide mouse input for each specific visualization with a

(a) A simple physical mouse.



(b) The same physical mouse being held by a right-handed person.



(c) The virtual mouse being "held" by a right-handed person.

Figure 4.1: A physical mouse versus the virtual mouse.

linked tool, and also allows simultaneous input to several visualizations concurrently.

Looking at the guidelines, the primary focus is on visualization interaction and therefore the virtual tool must always be available to be able to provide mouse input to the visualizations at all times. Therefore, the virtual tool is placed on top of the visualizations. Consequently, to not distract someone in providing input to the visualizations, the tool must be as dormant as possible when not in use. To enhance visibility of the virtual mouse, all elements of the mouse tool have a thin contrast border (both when the virtual mouse is in use and not used).

In the rest of this chapter the details for the design of the virtual tool are described. First, Section 4.1 provides the anatomy of the tool. Then, Section 4.2 explains the different movement modes the virtual tool provides. Lastly, Section 4.3 introduces the lens functionality the tool provides.

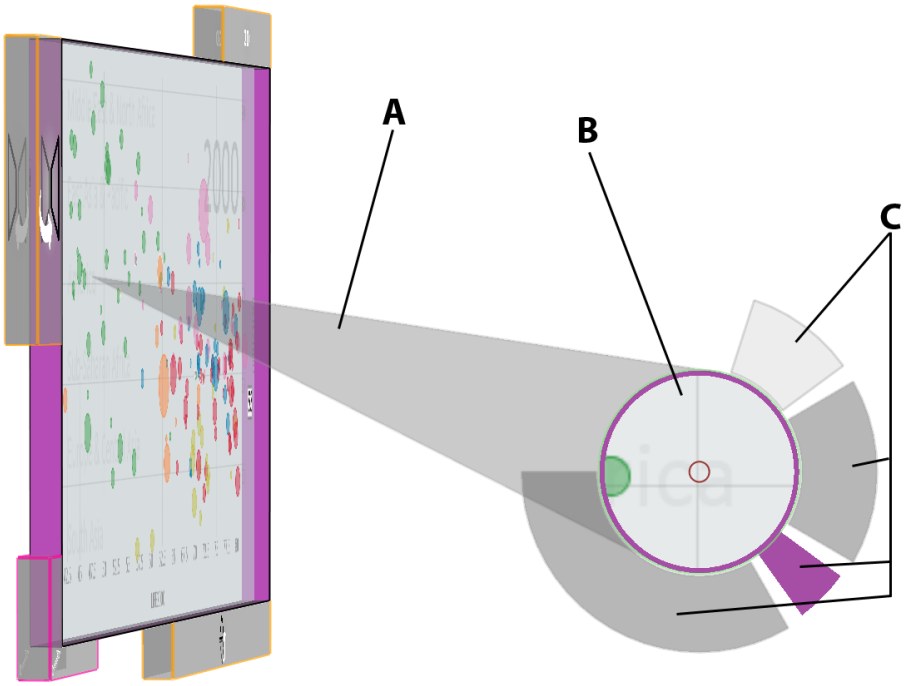


Figure 4.2: The basic layout for the virtual mouse. There is the base that is "held" (B), there are several wedges (C), and a cone pointing towards the actual mouse cursor position (A).

4.1 Anatomy of the tool

The virtual mouse (virtual tool) consists of several parts which can be divided up into three main interaction areas (see Figure 4.2). There is the base of the tool (B), which is designed as a circle, a uniform shape so all other elements have the possibility to adjust to any orientation. The base functions as the element that is "held" during interaction with the tool, like the base of a real mouse where the palm rests. There also is the cone (A) which points to the mouse cursor position on the visualization panel. Lastly, there are several wedges (C) which provide the mouse button interaction and configure parameters of the mouse tool itself.

A touch activates the mouse tool. On activation, as a visual cue to which mouse is being activated, the virtual mouse grows slightly, the colours of the mouse get brighter and the cone and buttons appear. The mouse can also be activated by touching the corresponding visualization panel. When the virtual mouse is deactivated, the mouse tool quickly shrinks in size again, the colours dim, and the cone and buttons disappear.

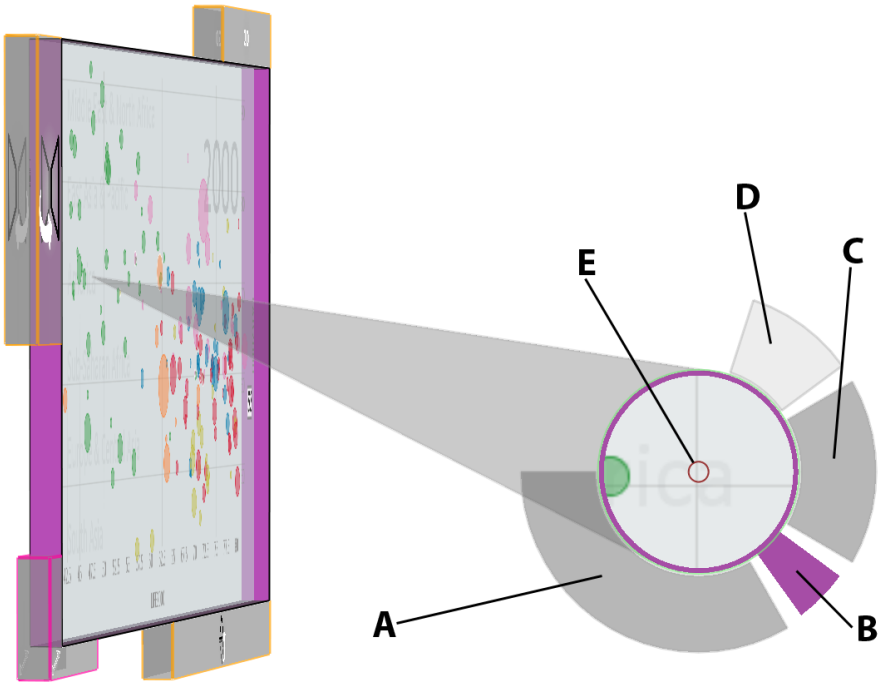


Figure 4.3: The detailed layout for the virtual mouse. There is the left- and right mouse button (A and C), and a zoom button (D). Furthermore, there is a colored wedge to provide an extension for the base of the tool (B), and a red circle indicating the mouse cursor position on the visualization.

4.1.1 Cone

The cursor of the virtual mouse (region A, see Figure 4.2) is designed as a cone. The cone is pointing to the cursor position on the visualization and has the circle as a "base". The cone provides a visual connection between the mouse cursor on the visualization and the mouse tool base and can be drawn transparently to minimize occlusion of the visualization. With its pointing shape, the cone also gives a visual cue to locate the mouse pointer when looking at the virtual mouse. The cone is designed to be (slightly) visible in all conditions as long as the mosue tool is active. The cone has a border color which has a high contrast with the fill color of the cone. The cone is only shown when the virtual mouse is activated; as soon as the virtual mouse is deactivated the cone begins to fade. After a few seconds the cone is completely transparent, and will remain inactive until the mouse is reactivated.

4.1.2 Buttons

During the design phase, I chose to design a two-button mouse, because emulating the third scroll wheel button of a physical mouse has not yet been defined for touch

interaction. For example, it is not well defined how to translate movement on a touch screen to the discrete steps a scroll wheel makes: how sensitive should the touch screen be? Currently the virtual mouse tool is a two-button virtual mouse, with the question of a third scroll wheel button left for future work.

The two mouse buttons of the virtual mouse are two wedges located around the base of the tool (parts A and C, see Figure 4.3). The current design is aimed for a right-handed person; when using the tool, the hand of the interacting person is often rotated slightly counter-clockwise with respect to a side of the table. Because of this rotation, the left mouse-button (A) is larger and the right button (C) is more toward the top of the base. Also, the buttons emulate the full button states of a physical mouse, which makes dragging possible.

Initially, the two buttons of the virtual mouse are not shown to reduce occlusion of the visualizations. When the virtual mouse is activated, the buttons become available in a short fade-in animation. When either of the buttons is touched, it gives a visual cue of this by changing its color.

One of the goals is to allow someone to use it reflexively when possible. Since there is no tactile feedback on touch screens yet, I chose to make the positions of the buttons fixed with respect to the base; they do not rotate with the rotation of someone's hand. This requires the person to always be on one side of the touch screen. It does, however, enable someone to, after training, be able to use the tool reflexively, as soon as the mouse is "grabbed".

Another thing that was considered was making the tool usable from several sides to allow collaboration from several sides, since the round base would easily allow rotation of the wedges around it. There has already been quite some research into automatic orientation on digital tables [Shen et al., 2004, Kruger et al., 2004]. In [Kruger et al., 2004], several design implications are provided to allow orientation on a tabletop, such as: "free rotation must be supported," "rotation techniques must be lightweight," and some more. However, when the tool would allow free rotation (and being lightweight) it becomes more difficult to use the mouse tool reflexively. This is mainly because automatic rotation according to someone's hand is difficult: the orientation has to be estimated from a few touch locations. Therefore, making the mouse tool usable from several sides was left as future work.

4.1.3 Identification

Linking visualization panel and virtual mouse together has advantages such as the ability to provide mouse input to several visualizations concurrently. However, there is also a downside: which virtual mouse belongs to which visualization panel?

To mediate this problem, several adjustments to the initial design were made. First, a unique color was added per linked visualization panel / virtual mouse pair. This

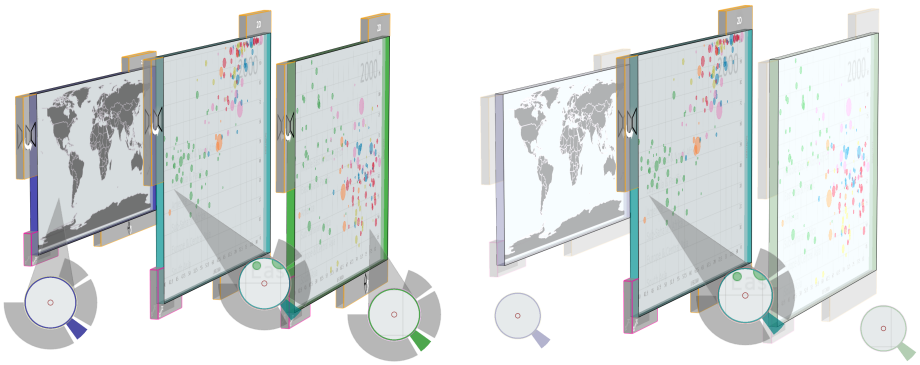


Figure 4.4: The two ways to identify which virtual tool belongs to which panel: each virtual tool and corresponding panel have a unique color, and on activation of a panel or corresponding virtual tool the other panels temporarily fade out.

unique color is shown on both the frame of the visualization panel and the rim of the virtual mouse (see Figure 4.4a). The wedge between the two buttons (part B, see Figure 4.3) also has this unique color. Upon activation, these colors become more pronounced by making the colors more opaque, thereby making it easier to see which mouse belongs to which panel.

Another addition to verify which virtual mouse belongs to which visualization panel is a short animation (1500ms) that slightly fades all panels except the panel that is interacted with upon activation of the visualization panel (see Figure 4.4b). This is most useful when the panels are viewed head-on because then the unique color of the panel on the frame is not visible.

4.2 Movement

To move the cursor on the visualization panel, the virtual mouse provides several ways: relative pointing, offset pointing, and pantograph pointing. Each of these methods is discussed in a section below. With all of these methods the cursor can never leave the visualization panel: when the cursor reaches the border and the pointing method wants it to move further, it will simply stay at the boundary.

4.2.1 Relative Pointing

Relative pointing is achieved by touch-dragging the base of the virtual mouse. This attaches the base to the finger, and also changes the cursor position on the visualization panel by moving it in the same direction as the base (with respect to the touch screen). To provide precision movement, here a control-display gain is used. When moving the finger fast, the cursor on the visualization panel moves at the same speed as the finger.

With slow movements, the cursor moves several times slower on the visualization panel. The control-display gain is calculated by looking at the last 500ms of movement of the virtual mouse. The display gain is calculated with the following formula (m is the amount of pixels moved by the finger):

$$scale = min(1, m * 0.007)$$

Using this display gain factor is similar to what can be achieved with a regular mouse; in most operating systems the pointer speed can be chosen to allow for the rough movements to traverse big parts of the screen. Also, most operating systems have some option to "enhance precision" in which they slow down cursor movement when the mouse is moving slowly. While my technique does not provide a configurable pointer speed, it does provide the enhanced precision option which takes into account the movement speed of the virtual mouse.

4.2.2 Offset Pointing

Offset pointing is started by interacting with the narrowest third of the tip of the cone. Here the interaction is similar to Baudisch et al. [2003], except that the base of the virtual mouse is also held. With this pointing method, the actual cursor position is determined solely by looking at the position of the finger touching the cone, and the direction the cone makes. This position is then offset and used as cursor position for the visualization.

4.2.3 Pantograph Pointing

Pantograph pointing is started by interacting with the widest two-thirds of the cone area. Then, the relationship of the distances between the cursor and cone touch (d1) and the cone touch and lens touch (d2) is preserved: $\frac{d1}{d2} = const = \frac{d1'}{d2'}$ (see Figure 4.5). This method is similar to the pantograph technique Collomb et al. [2005] but uses two touches (like the two-handed interaction technique in Abednego et al. [2009]). This way of pointing allows people to comfortably interact with distant objects.

4.3 Lens

Initially, the visualizations also provided a static thumbnail to be shown on the base of the virtual mouse as an extra visual cue to visually link mouse and panel together. However, I realized that the last part of the visualization seen by a person would most likely be located at the mouse cursor position of the visualization. Therefore, I used a focused part of the visualization as a thumbnail. However, this can cause problems when the part that was last interacted with in the visualization has almost no details

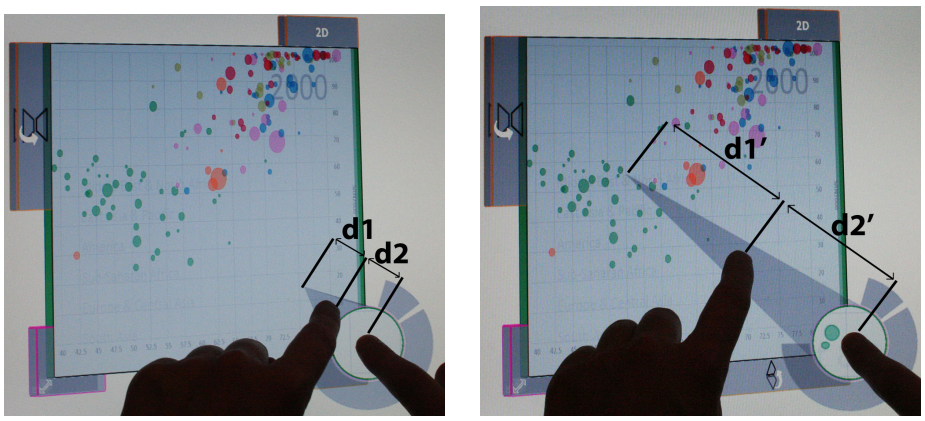


Figure 4.5: Pantograph pointing with the virtual mouse. The relationship of the distances between the cursor and cone touch (d1) and the cone touch and lens touch (d2) is preserved: $\frac{d1}{d2} = const = \frac{d1'}{d2'}$.

in it. This occurs, for example, if the visualization consists of a homogeneous color around the cursor.

After using this localized thumbnail, I discovered that the thumbnail on the base was more than just a thumbnail: it provides a flat view of the visualization and lens functionality. The lens provides the ability to still do precise pointing when the actual visualization is shown skewed on the display (see Figure 4.6). There is also work by Nacenta et al. [2007], who performed a study to look at the viewing angle of a person with respect to the display, and whether a perspective-aware interface improves results for operations like targeting. Although this does not relate directly to the skewed visualization and the "flat" view provided by the virtual mouse, this can be taken as a suggestion that providing a flat view of (part of) the visualization can help with targeting. However, it does add some overhead for the person interacting with the system, since they have to relate the view of the skewed visualization to the flat view of the visualization in the lens. To be able to actually use this lens and not occlude the lens itself when interacting with the virtual mouse, a handle was added to the base of the virtual mouse (part B, see Figure 4.3). This handle can be used just like the base itself, it only this prevents occlusion of the lens.

To address the problem of a non-representative thumbnail and to give people the ability to actually adjust the size of the virtual mouse, a button and a two-touch gesture were added. With two touches on the base of the tool, the base of the tool can be resized with the well known pinching gesture. The extra button (part D, see Figure 4.3) allows control over the zoom of the lens in a way a DSLR lens zooms. The button can be rotated clockwise and counter clockwise to respectively zoom in and out.

During test sessions of using the virtual mouse, it was discovered that, when a

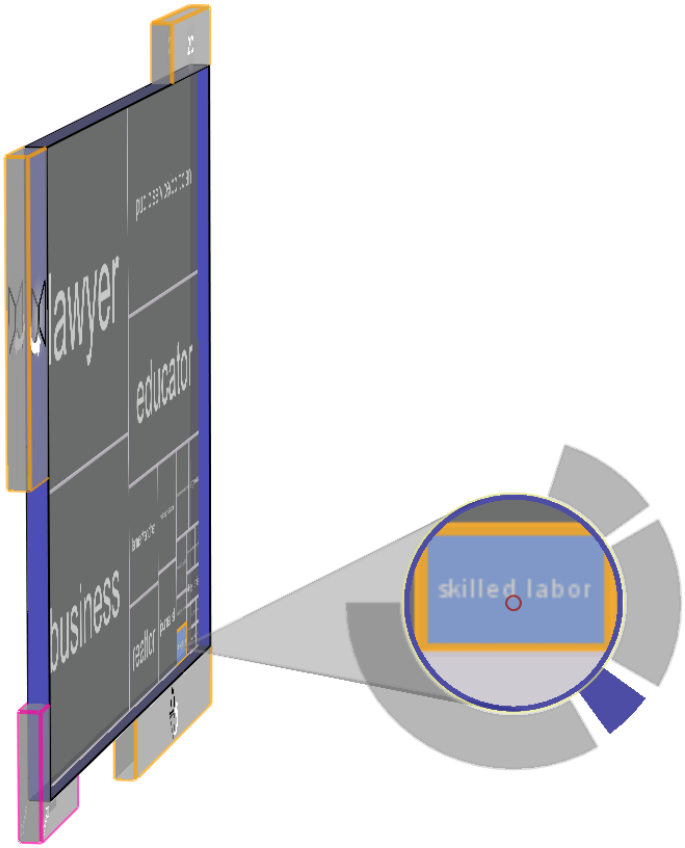


Figure 4.6: Virtual mouse interaction with a skewed panel. Since the base region provides a flat view of the visualization "window" the interacting person is still able to provide precise input to the visualization.

person was using the lens to point precisely, it was difficult to determine where the exact position of the cursor was. This was because it was difficult to determine the exact center of the lens. Because of this, a red circle is added inside the base (part E, see Figure 4.3), providing an indication of where the cursor is on the visualization.

4.4 Summary

In this chapter a design is proposed to provide mouse input via MT input. A two-button virtual mouse tool is created that is able to provide relative and absolute mouse input to a visualization. The mouse tool has a round base, two buttons, a cursor pointing to the actual cursor position on the visualization panel, and a lens that provides a flat localized view of the visualization. Moreover, there is a button to control the zoom level of the lens.

Chapter 5

Discussion

In this chapter, the first section provides a short review on implementation issues that affected certain design decisions and other usability measures. Then, the second section provides an application scenario that outlines a possible use case for my design. Thereafter, an evaluation is provided that gives some initial feedback from possible users of the system.

5.1 Implementation

The program created during my research is based on the VisLink program by Collins and Carpendale [2007] and a framework (MT3D) created by Hancock for his research for, among others, Hancock et al. [2007, 2009]. The framework by Hancock provides both a 3D OpenGL environment and easy access to multi-touch capabilities, whereas the VisLink program by Collins creates interplanar edges and allows me to use existing prefuse visualizations in 3D. I merged these two projects into one, and used that as a base to create the interaction technique presented in this thesis.

5.1.1 Merging the Projects

In merging both projects, an advantage was that both Vislink and the MT3D framework use Java as a programming language. However, both were standalone programs. In the end I ported the VisLink code to the MT3D framework, since this seemed to be the easiest solution. Unfortunately, most of the code from VisLink had to be changed quite rigorously, and the link between prefuse and VisLink had to be rewritten almost completely. Still, by using the MT3D framework as a base, using MT was easy since the framework uses an event based setup where events are issued when a 3D object is touched. Moreover, the Sticky Tools technique from Hancock et al. [2009] was available immediately. This then saved much time because prototyping

MT interactions was relatively easy now.

In order to import a visualization into the regular VisLink program, a prefuse visualization had to be rewritten partly. Moreover, the loading of the data for the interplanar edges between visualizations in VisLink was completely static and slow. Therefore, I improved on the way existing visualizations could be imported into the program by creating a simple interface *VisLinkInterface* that has to be implemented, together with a default implementation of that interface (*VisLinkInterfaceAdapter*). When using the default *VisLinkInterfaceAdapter* implementation, an existing prefuse visualization only has to extend that class instead of the default prefuse display class and the visualization should work. However, this does only provide the normal VisLink functionality: when something extra is required the visualization would have to override certain functions itself. For example, with a world map visualization it could be useful to, when a datapoint is selected, also select the neighboring points at a lower activation level.

Furthermore, loading of the interplanar edges data was improved by having the data specified in a XML file, and having the interplanar edges data automatically created from that. Within the program, the speed of loading of the interplanar edges data was improved on by putting all the data of the visualization in a map beforehand, and then using that as a lookup table instead of querying the visualization each time a datapoint was required.

5.1.2 Performance Issues

During the design of the interaction technique itself, many performance issues arose when testing the prototype on a MT table like the Microsoft Surface or the Smart Table. One of the causes of the performance problems was that the computers in these tables are relatively slow compared to a desktop pc. The parts of the program that required most attention were parts where geometry of the elements on the screen could not be cached because the size of that element could change at every frame. Some of these elements also required much computation, for example the circles or parts of circles required to draw the virtual mice. Here, the interesting part was that the *sin* and *cos* functions were extremely slow on the MT tables in comparison to a desktop pc. Therefore, for the elements that could not be cached, I cached all the geometry information as generic as possible, and then would only compute scale factors. For example, I stored a generalized form of a circle in an array instead of calling the *sin* and *cos* functions for each rendered frame.

Another big performance bottleneck was the rendering of the interplanar edges In the regular VisLink program, the interplanar edges would be recomputed each frame. At first I just included this in the program as it was implemented in the regular VisLink program, but quickly found out that, as soon as there were more than a few links,

the performance would drop to less than one frame per second on the MT tables. Therefore, I created compiled versions of each interplanar edge, and rendered those, restoring performance to 60fps again. Moreover, the propagation of the edges was moved to a separate thread, thereby preventing a complete lockdown of the program but instead still allowing user interaction while the edges would propagate. When the propagation of edges still ran in the graphics rendering thread, the program would otherwise freeze for a few seconds when a set of 1000 or more interplanar edges would be activated.

A last big performance bottleneck was reshaping of the visualization. This was caused by having the visualization redrawn each time the panel would reshape a small bit over the course over the reshaping action, thereby effectively freezing the application until the reshaping action finished. By instead using a timeout function to only redraw the visualization once the reshaping action finished, the performance would only drop for less than a second after the action finished.

5.1.3 Algorithmic Problems

During the development of the interaction technique, there were also problems which could not be solved by adding a cache somewhere. One of these issues is the ordering of the panels in 3D space, and thus between which pairs of panels interplanar edges should be drawn. When the panels are allowed 6DOF movement this is not easy to solve. One could apply a travelling salesman problem algorithm to this so that the panels are ordered by a minimum travelled path. However, this algorithm does not, for example, take into account the orientation of the panel. In my program, the panels are simply ordered by taking an order along one axis. This does, however, sometimes provide problems, as can be seen in the scenario in the next section; the grid layout orders the interplanar edges depending on the y position of the panel.

Another problem that posed itself frequently was the math used to determine from which side a panel was viewed. Knowing the viewed side of the panel is important to display the visualization in the correct way. However, even in the final version, the method to determine the viewed side is not entirely correct yet; when looking at the garage view the shown side of the visualization is sometimes incorrect. This is caused by having the rotation and translation matrix separate, and thus having difficulties reconstructing the viewport.

5.2 Application Scenario

In this section a scenario is described to demonstrate how my interaction design can improve a data analysis process where a quite diverse dataset is used with several information visualization techniques applied to it. The people work together using

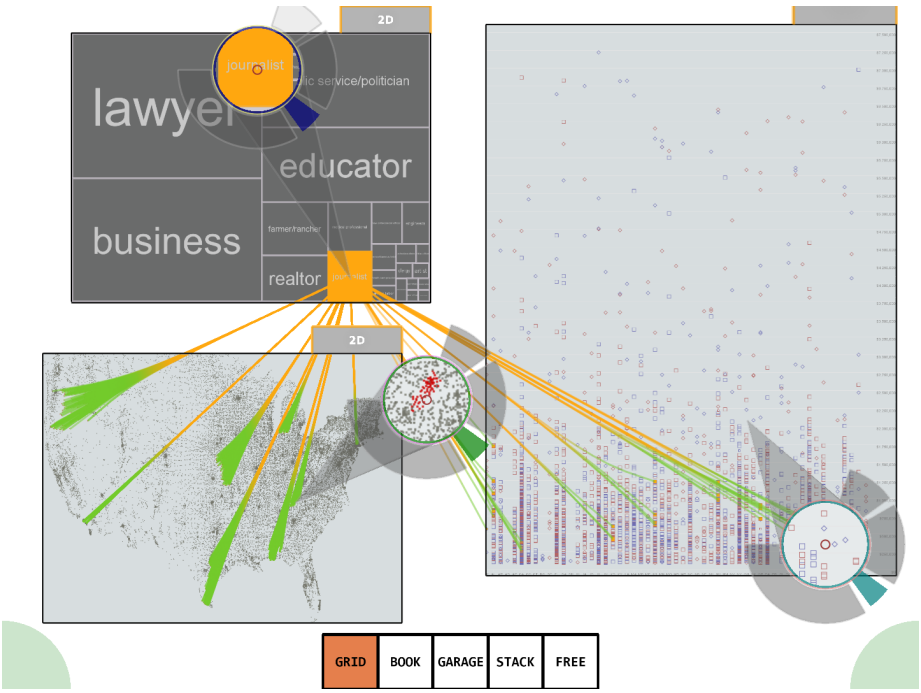


Figure 5.1: The visualizations in grid-layout so each of the analysts can work on a separate visualization in a flat view.

a MT table running my program to do this analysis process, since it allows them to work both concurrently on different tasks, as well as doing joint work when required.

The scenario described follows three data analysts, Andy, Bob and Christine, through a data analysis session. They are asked to find some juicy facts for a newspaper about electoral candidates in the USA.

Before the group meets, Bob imports the dataset into the program. The dataset contains information on how much funding members of political parties raised for their parties, as well as what their jobs are and where these members live. When Andy and Christine join Bob, they decide to use three visualizations to visualize this data; one visualization with a map for the geographical location, a treemap visualization to show the occupation of the political party members, and a scatterplot visualization showing states and funding (see Figure 5.1).

Andy then switches to grid layout (see Figure 5.1), to get a flat view on all the visualization, so each of them can work on a separate visualization. Christine sees the text "journalist" on the treemap visualization, and uses a virtual mouse to select all journalists, thereby creating interplanar edges showing all journalists on the other visualizations (see Figure 5.2).

The data analysts then each start exploring a different dataset. Christine explores

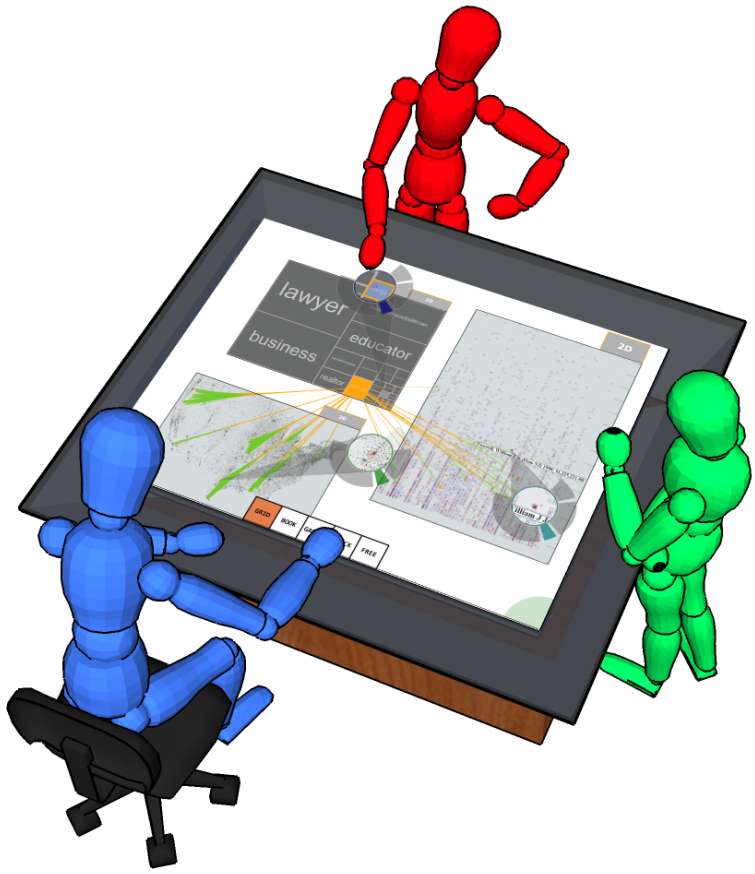


Figure 5.2: Andy (blue) switches to grid layout, and Christine (red) selects the "journalist" node in the treemap visualization using a virtual mouse.

the scatterplot, looking at the datapoints by hovering over them with the virtual mouse, thereby getting tooltips providing extra information on the data. Meanwhile, Andy explores the edge bundles on the geographical map, trying to find information there. Note that Christine and Andy can work simultaneously because they use separate virtual mice.

After some time, they conclude they have not found any juicy facts yet. Therefore, Bob switches to stack layout. Bob then says, "Look at this!" and points at a bundle between the scatterplot visualization and geographical map visualization. Christine sees it too, but Andy asks Bob to explain. Therefore, Bob switches to the book layout to better emphasize the interplanar edges, and drags the panels apart using two hands (see Figure 5.3). He then explains to Andy that journalists in the south of the USA raise considerably less money for their parties.

In this scenario the concurrent use of virtual mice worked well, since Christine and

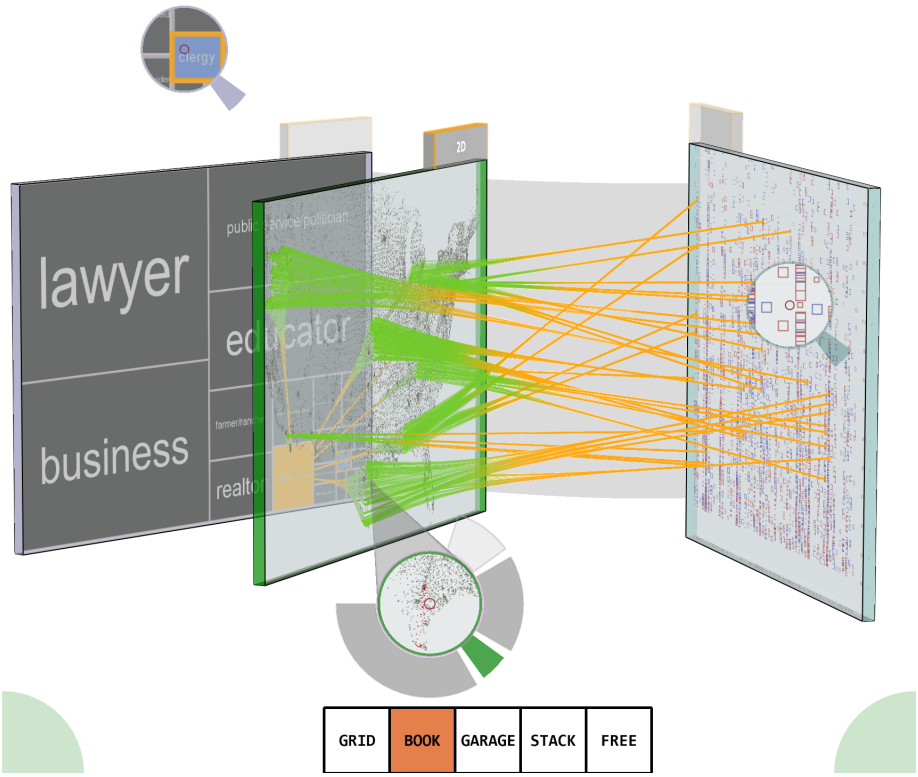


Figure 5.3: Bob switched to the book layout, showing that journalists in the south of the USA raise considerably less money.

Andy could work concurrently on two visualizations. Also the MT gesture to move panels in the book layout worked well, since a simple two-handed gesture could be performed to move them apart. However, the disjoint work using the grid layout could be better, since using a virtual mouse to explore a scatterplot by hovering over all the datapoints is quite cumbersome. Moreover, the interplanar edges are not visualized well in the grid layout since they are sometimes on top of the visualization and other times they are not.

5.3 Participatory Design Process

During the whole development of the interaction technique, there were several brainstorm and feedback sessions. During these sessions, sometimes only a few people, but other times most of the lab participated in providing useful comments on drawbacks or positive parts of the interaction technique. During the design, some concessions had to be made to make it work in most situations. For example, initially the cone of the virtual mouse was always visible, and faded when the mouse was inactive for

some time. During test sessions later, however, it was discovered that when trying to work with the visualizations themselves a lot of accidental touches on the mouse cursors would occur. Therefore, it was decided that the cursor completely disappears after approximately a second. During another test session with the whole lab, it was discovered that it was difficult to identify which mouse should be used to interact with a certain visualization panel. Therefore, more identification methods were introduced to link the virtual mouse and panel together, for example, the temporary fading of the other panels. Many more of these design changes occurred during the whole design phase; in the end making it a better technique where most people can work with.

However, there are known problems with the current interaction technique. The most important problem is that the whole user interface and design of the program is assuming there is a single side which is the "bottom" of the screen. The interface elements are also facing in one direction, and elements cannot always be rotated towards another side. The most problematic points with this is that the layouts all assume a certain direction. For example, the grid layout does not allow a panel to be rotated at all, instead the two-touch gesture is mapped to reshaping. The same applies to the book, garage and stack view. Here the path the panels follow cannot be rotated to suit a person's view point. This problem also presents itself with the virtual mouse. Here we made the design choice to not rotate the buttons to allow reflexive use of the system when one has become an expert. This does, however, imply that one cannot use the mouse from any other side of the table. A possible solution to this problem might be an element which can be dragged along the corners of the screen, and thereby changing the bottom side of the screen. However, this is a problem that has not been solved yet completely in the literature and therefore might be very difficult to fix in the program.

Another issue that caused problems for people was the sudden switch when changing layouts. Here an animation showing the movement of the panels from one layout to another might be a good improvement. However, due to time constraints this was not implemented.

5.4 Informal Evaluation

Due to time constraints, unfortunately it was not possible to conduct a formal controlled study on how well the virtual mouse can be used, or if the tandem with the 3D visualization interaction works well. However, in the mean time an informal study was done in Ontario at UOIT with a set of undergraduate human-computer interaction students [Vlaming et al., 2010]. One photo of someone working with the system can be seen in Figure 5.4.

When the students started working with the demo, most surprisingly was the fact that the virtual mouse was seen as lens to examine the data and not as a virtual mouse

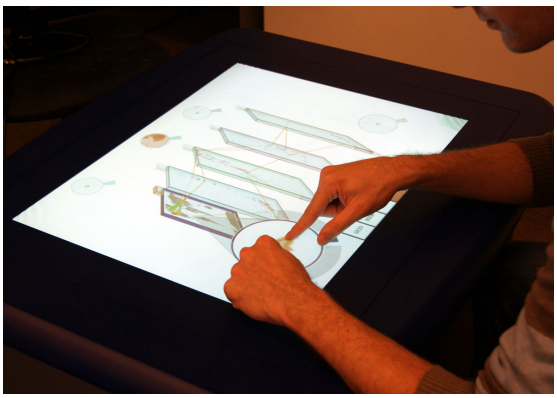


Figure 5.4: A picture of the system in actual use.

pointing towards a visualization. They did, however, use all the basic functionality the virtual mouse provided such as dragging and clicking.

Another surprising fact was the almost immediate ability to use direct pointing by interacting with both the cone and base of the virtual mouse. Moreover, several students used the direct pointing technique exclusively. These students then had problems selecting small targets on the visualizations. This could have been caused by the minimal instruction that was given or by a design flaw in the precise pointing technique making that technique unfavorable, or even the visual design itself.

During the test session with the demo, to explore the provided dataset the students visited and reused all layouts that were provided. However, they did seem to prefer the bookview the most to compare visualizations with each other. To decide which visualization to look at next the grid layout was used most often.

While the program does allow concurrent use of the system by several people, during the test session there was no concurrent use at all. This may be due to the physical size of the table since a Smart Table was used, which is designed to be used by children. Another possibility is that the design of the controls is flawed since they do imply a preferred side of the table.

Chapter 6

Conclusion

In this thesis I designed and developed methods to use visualizations on a touch screen. My main contribution is a virtual mouse as a whole, which lives on-top of the application and reaches to its connected visualization with a cone. The virtual mouse is designed to visually resemble a real mouse for recognizability and uses MT as input. Because there is one virtual mouse per visualization, people can work with several visualizations concurrently. Moreover, the virtual mouse allows both relative and absolute pointing, thereby providing the best of both worlds: precision and relative pointing as provided by a regular mouse, and direct pointing as is provided by most MT interaction techniques. Furthermore, by emulating mouse input via a tool, all visualizations written in the popular prefuse toolkit can be used immediately in my system, without requirements to completely re-implement the visualizations.

The virtual mouse works in tandem with an 3D interaction set to allow 3D manipulation of these visualizations. As a group, this 3D interaction set is my second contribution; the layouts are the most important part of this contribution. As a third contribution I created a new way to provide precise input on touch screens.

However, as with all work, there are also some areas in the solutions provided that provide possibilities for future work.

Layouts For the panel layouts, one of the things that could be useful is the ability to have layout slots instead of fixed buttons to switch between the layouts, allowing the person to have multiple layouts of the same type. Moreover, performing a controlled user evaluation on the already provided layouts would be useful, validating their use and existence, and maybe extending the list of layouts available. This could then also show if the modes that are introduced with the layouts give problems when the application is used in a collaborative setting. Moreover, a history of what happened in each layout could be useful in visualization exploration. This could be provided by remembering key moments during the exploration process. Lastly, an animation when

a person switches between layouts could be good for retaining visual consistency.

Virtual Mouse For the virtual mouse, a useful feature would be the adaptation to a left or right-handed person. Whether or not a person is left or right-handed could be determined by the orientation of the touch. When the tool is used for longer periods with a left-handed orientation, the tool could switch to left-handed orientation and vice versa, though there would be a need for some notification. Another possibility would be to have a switch on the tool that controls left or right-handed modes. Moreover, performing a controlled user evaluation on the use of the virtual mouse could provide much insight, thereby testing whether the virtual mouse is actually a viable method of providing mouse input on a touch screen. This could also give insight in how much performance is lost by using the virtual mouse instead of a regular mouse to provide mouse input. It would be interesting to see if this performance loss can outweigh gains such as easier interaction in a collaborative setting.

Bibliography

M. Abednego, J.-H. Lee, W. Moon, and J.-H. Park. I-Grabber: expanding physical reach in a large-display tabletop environment through the use of a virtual grabber. In *Proc. of the Int. Conf. on Interactive Tabletops and Surfaces*, pages 61–64, New York, 2009. ACM. ISBN 978-1-60558-733-2. doi: http://doi.acm.org/10.1145/1731903.1731917. Cited on page 55.

A. Agarawala and R. Balakrishnan. Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1283–1292, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7. doi: http://doi.acm.org/10.1145/1124772.1124965. Cited on pages 25 and 26.

P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and A. Zierlinger. Drag-and-Pop and Drag-and-Pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. IFIP World Computer Congress*, pages 57–64, 2003. URL `http://research.microsoft.com/en-us/um/people/cutrell/DragAndPop_Interact2003.pdf`. Cited on page 55.

R. Bencina and M. Kaltenbrunner. The design and evolution of fiducials for the reactivision system. In *Proceedings of the 3rd International Conference on Generative Systems in the Electronic Arts (3rd Iteration 2005)*, Melbourne, Australia, 2005. Cited on page 11.

H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1263–1272, New York, NY, USA, 2006. ACM. ISBN 1-59593-372-7. doi: http://doi.acm.org/10.1145/1124772.1124963. Cited on page 30.

F. Bérard and Y. Laurillau. Single user multitouch on the diamondtouch: from 2 x 1d to 2d. In *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 1–8, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-733-2. doi: http://doi.acm.org/10.1145/1731903.1731905. Cited on page 17.

D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004. ISBN 0201758679. Cited on page 40.

S. K. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think (Interactive Technologies)*. Morgan Kaufmann, 1st edition, February 1999. ISBN 1558605339. Cited on page 1.

C. Collins and S. Carpendale. Vislink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2007.70611. Cited on pages 1, 2, 3, 26, and 59.

M. Collomb, M. Hascoët, P. Baudisch, and B. Lee. Improving drag-and-drop on wall-size displays. In *Proc. of Graphics Interface*, pages 25–32, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society. ISBN 1-56881-265-5. URL `http://portal.acm.org/citation.cfm?id=1089514`. Cited on page 55.

P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM. ISBN 1-58113-438-X. doi: http://doi.acm.org/10.1145/502348.502389. Cited on page 18.

A. Esenther and K. Ryall. Fluid dtmouse: better mouse support for touch-based interactions. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 112–115, New York, NY, USA, 2006. ACM. ISBN 1-59593-353-0. doi: http://doi.acm.org/10.1145/1133265.1133289. Cited on page 28.

J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM. ISBN 1-59593-271-2. doi: http://doi.acm.org/10.1145/1095034.1095054. Cited on page 11.

M. Hancock, S. Carpendale, and A. Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two- and three-touch techniques. In *Proc. CHI 2007*, pages 1147–1156, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-593-9. doi: http://doi.acm.org/10.1145/1240624.1240798. (Nominated for Best Paper Award). Cited on pages 20 and 59.

M. Hancock, T. ten Cate, and S. Carpendale. Sticky tools: Full 6DOF force-based interaction for multi-touch tables. In *Proceedings of the ACM International*

Conference on Interactive Tabletops and Surfaces (ITS), pages 145–152, New York, NY, USA, 2009. ACM Press. Cited on pages 6, 20, 21, 41, and 59.

M. S. Hancock, S. Carpendale, F. D. Vernier, D. Wigdor, and C. Shen. Rotation and translation mechanisms for tabletop interaction. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 79–88, Washington, DC, USA, 2006. IEEE Computer Society. ISBN 0-7695-2494-X. doi: http://dx.doi.org/10.1109/TABLETOP.2006.26. Cited on page 20.

J. Heer, S. K. Card, and J. A. Landay. prefuse: A toolkit for interactive information visualization. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, pages 421–430. ACM, 2005. doi: http://doi.acm.org/10.1145/1054972.1055031. Cited on page 2.

O. Hilliges, D. Kim, and I. Izadi. Creating Malleable Interactive Surfaces using Liquid Displacement Sensing. In *In Proceedings of the 3rd IEEE Tabletop and Interactive Surfaces, Amsterdam, the Netherlands*, Oct. 2008. Cited on page 13.

O. Hilliges, S. Izadi, A. D. Wilson, S. Hodges, A. Garcia-Mendoza, and A. Butz. Interactions in the air: adding further depth to interactive tabletops. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 139–148, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-745-5. doi: http://doi.acm.org/10.1145/1622176.1622203. Cited on page 23.

E. S. Hong, H. Zhang, and Y. sheng Guan. Sensing contact with analog resistive technology, 1999. Cited on page 16.

E. Hornecker, P. Marshall, N. S. Dalton, and Y. Rogers. Collaboration and interference: awareness with mice or touch input. In *CSCW '08: Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 167–176, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-007-4. doi: http://doi.acm.org/10.1145/1460563.1460589. Cited on page 2.

P. Isenberg and D. Fisher. Collaborative brushing and linking for co-located visual analytics of document collections. In *Computer Graphics Forum*, volume 28, pages 1031–1038. John Wiley & Sons, 2009. Cited on pages 5, 33, and 34.

P. Isenberg, A. Tang, and S. Carpendale. An exploratory study of visual information analysis. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1217–1226. ACM, 2008. Cited on page 5.

P. Isenberg, A. Bezerianos, N. Henry, S. Carpendale, and J.-D. Fekete. Coconuttrix: Collaborative retrofitting for information visualization. *IEEE Comput. Graph. Appl.*,

29(5):44–57, 2009. ISSN 0272-1716. doi: http://dx.doi.org/10.1109/MCG.2009.78. Cited on pages 5 and 33.

D. H. Jeong, W. Ribarsky, and R. Chang. Designing a pca-based collaborative visual analytics system, 2009. Cited on page 32.

K. Kin, M. Agrawala, and T. DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *GI '09: Proceedings of Graphics Interface 2009*, pages 119–124, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society. ISBN 978-1-56881-470-4. Cited on page 2.

R. Kruger, S. Carpendale, S. D. Scott, and S. Greenberg. Roles of orientation in tabletop collaboration: Comprehension, coordination and communication. *Journal of Computer Supported Collaborative Work*, 13(5-6):501–537, 2004. doi: http://dx.doi.org/10.1007/s10606-004-5062-8. Cited on page 53.

R. Kruger, S. Carpendale, S. D. Scott, and A. Tang. Fluid integration of rotation and translation. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 601–610, New York, NY, USA, 2005. ACM. ISBN 1-58113-998-5. doi: http://doi.acm.org/10.1145/1054972.1055055. Cited on page 19.

A. Lex, M. Streit, E. Kruijff, and D. Schmalstieg. Caleydo: Design and evaluation of a visual analysis framework for gene expression data in its biological context. In *Proc. of the IEEE Pacific Visualization Symp.*, pages 57–64. IEEE, 2010. doi: http://dx.doi.org/10.1109/PACIFICVIS.2010.5429609. Cited on page 27.

J. Light and J. Miller. Miramar: a 3d workplace. In *Professional Communication Conference, 2002. IPCC 2002. Proceedings. IEEE International*, pages 271–282, 2002. doi: 10.1109/IPCC.2002.1049110. Cited on page 24.

J. Matejka, T. Grossman, J. Lo, and G. Fitzmaurice. The design and evaluation of multi-finger mouse emulation techniques. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1073–1082, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7. doi: http://doi.acm.org/10.1145/1518701.1518865. Cited on page 29.

Microsoft. What is the touch pointer? Retrieved Feb 08 2010, URL `http://windows.microsoft.com/en-US/windows-vista/What-is-the-touch-pointer`, 2010. Cited on page 29.

M. A. Nacenta, S. Sakurai, T. Yamaguchi, Y. Miki, Y. Itoh, Y. Kitamura, S. Subramanian, and C. Gutwin. E-conic: a perspective-aware interface for multi-display environments. In *UIST '07: Proceedings of the 20th annual ACM symposium*

on User interface software and technology, pages 279–288, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-679-2. doi: http://doi.acm.org/10.1145/1294211.1294260. Cited on page 56.

M. A. Nacenta, P. Baudisch, H. Benko, and A. Wilson. Separability of spatial manipulations in multi-touch interfaces. In *GI '09: Proceedings of Graphics Interface 2009*, pages 175–182, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society. ISBN 978-1-56881-470-4. Cited on page 41.

J. L. Reisman, P. L. Davidson, and J. Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 69–78, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-745-5. doi: http://doi.acm.org/10.1145/1622176.1622190. Cited on pages 22 and 43.

G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The task gallery: a 3d window manager. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 494–501, New York, NY, USA, 2000. ACM. ISBN 1-58113-216-6. doi: http://doi.acm.org/10.1145/332040.332482. Cited on page 27.

I. Rosenberg and K. Perlin. The unmousepad: an interpolating multi-touch force-sensing input pad. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–9, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi: http://doi.acm.org/10.1145/1576246.1531371. Cited on page 16.

C. Shen, F. D. Vernier, C. Forlines, and M. Ringel. Diamondspin: an extensible toolkit for around-the-table interaction. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 167–174, New York, NY, USA, 2004. ACM. ISBN 1-58113-702-8. doi: http://doi.acm.org/10.1145/985692.985714. Cited on page 53.

Smart Technologies. Digital vision touch technology. Technical report, Smart, 2003. Retrieved Jan 07 2010, URL `http://smarttech.com/DViT/DViT_white_paper.pdf`. Cited on page 15.

Sun. Project looking glass. Retrieved Jan 13 2010, URL `http://www.sun.com/software/looking_glass/`, 2010. Cited on page 25.

T. ten Cate. A virtual sandtray on a tabletop computer. Master's thesis, RijksUniversiteit Groningen, 2009. Cited on pages 15 and 19.

M. Tobiasz, P. Isenberg, and S. Carpendale. Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization*

and Computer Graphics, 15:1065–1072, 2009. ISSN 1077-2626. doi: http://doi.ieeecomputersociety.org/10.1109/TVCG.2009.162. Cited on pages 5, 32, and 33.

E. Tse, J. Histon, S. Scott, and S. Greenberg. Avoiding interference: how people use spatial separation and partitioning in SDG workspaces. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 252–261. ACM, 2004. Cited on page 33.

L. Vlaming, C. Collins, M. Hancock, M. Nacenta, T. Isenberg, and S. Carpendale. Integrating 2d mouse emulation with 3d manipulation for visualizations on a multitouch table. *Proc. of the Int. Conf. on Interactive Tabletops and Surfaces*, 2010. To appear. Cited on pages I and 65.

D. Vogel and P. Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 657–666, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-593-9. doi: http://doi.acm.org/10.1145/1240624.1240727. Cited on page 31.

S. Voida, M. Tobiasz, J. Stromer, P. Isenberg, and S. Carpendale. Getting practical with interactive tabletop displays: Designing for dense data, "fat fingers," diverse interactions, and face-to-face collaboration. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, New York, NY, USA, 2009. ACM Press. Cited on pages 5, 6, and 31.

A. D. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk. Bringing physics to the surface. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 67–76, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-975-3. doi: http://doi.acm.org/10.1145/1449715.1449728. Cited on pages 22 and 23.

K. Yatani, K. Partridge, M. Bern, and M. W. Newman. Escape: a target selection technique using visually-cued gestures. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 285–294, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: http://doi.acm.org/10.1145/1357054.1357104. Cited on page 31.

J. S. Yi, Y. a. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007. ISSN 1077-2626. doi: http://dx.doi.org/10.1109/TVCG.2007.70515. Cited on page 4.