University of Groningen
Faculty of mathematics and natural sciences
Computing Science

Master of Science Thesis

Gesture Previews

by

Writser Pier Cleveringa

Supervisor: Tobias Isenberg

Groningen, 2009

To ...

Contents

Contents i

1 Introduction 3

2 Touch screen technology 7
- 2.1 Introduction . . . 7
- 2.2 Technology overview . . . 8
  - 2.2.1 Resistive touch screens . . . 8
  - 2.2.2 Capacitive touch screens . . . 9
  - 2.2.3 FTIR . . . 9
  - 2.2.4 Other technologies . . . 11
- 2.3 Conclusion . . . 12

3 Interaction on touch screens 13
- 3.1 Introduction . . . 13
- 3.2 Porting a traditional interface . . . 13
- 3.3 Novice versus expert users . . . 15
- 3.4 Cooperation and territoriality . . . 16
- 3.5 Conclusion . . . 18

4 Gestures 19
- 4.1 Introduction . . . 19
- 4.2 The definition of a gesture . . . 20
- 4.3 Multi-touch and cooperative gestures . . . 21
- 4.4 Local and global gestures . . . 22
- 4.5 Discussion . . . 24
- 4.6 Conclusion . . . 26

5 Gesture previews 27
- 5.1 Introduction . . . 27
- 5.2 The concept . . . 27

5.3 The overlay . . . 28
5.4 Composed gestures . . . 30
5.5 Multi-touch previews . . . 33
5.6 Conclusion . . . 33

6 Implementation **35**
6.1 Introduction . . . 35
6.2 Design Choices . . . 36
6.2.1 Gesture Storage . . . 36
6.2.2 Gesture detection algorithm . . . 37
6.2.3 Rotation and orientation . . . 38
6.3 Framework design . . . 40
6.4 Proof-of-concept application . . . 41
6.5 Gesture modes . . . 44
6.6 Conclusion . . . 47

7 User study **49**
7.1 Introduction . . . 49
7.2 Users study setup . . . 50
7.3 Data capturing and recording . . . 50
7.4 Analysis of video footage and questionnaire . . . 52
7.4.1 Gestural interfaces . . . 52
7.4.2 Gesture Previews . . . 53
7.4.3 Gesture drawing speed . . . 54
7.4.4 Composed Gestures . . . 54
7.4.5 Other observations . . . 55
7.5 Analysis of recorded gestures . . . 57
7.5.1 A word of warning . . . 57
7.5.2 The effect of gesture previews . . . 58
7.6 Conclusion . . . 61

8 Conclusion **63**
8.1 Introduction . . . 63
8.2 Interpretation of user study . . . 63
8.2.1 Gesture Previews for novice users . . . 63
8.2.2 Gesture Previews for expert users . . . 65
8.2.3 Increase of accuracy . . . 65
8.3 Future work . . . 66
8.4 Conclusion . . . 70

Bibliography **71**

A A sample gesture definition in XML **77**

B Gestures in the emergency control room **79**

C Gesture Previews Questionnaire **83**
C.1 Vragenlijst voorafgaand aan experiment. 83
C.2 Vragenlijst na experiment. 83

List of Figures **87**

Acknowledgements

Thanks to:

Tobias Isenberg, for giving valuable feedback about my thesis and for making sure it was actually finished. Arnoud de Jong, for being my mentor at TNO and for his enthousiast approach to everything regarding touch screens. Arnout de Vries, for valuable insights about user interfaces. Marco Aiello, for being my second supervisor. Arnaud, Anja and Maarten, for being nice roommates at TNO. Arjan Dijkstra, for proofreading this thesis. Everyone at TNO, for being enthousiastic colleagues, always eager to help. All participants of the user study, for their cooperation. Everyone else who helped me and I forgot to mention. And finally my friends and family, for supporting me and providing welcome distraction while I was working on my thesis.

Chapter 1

Introduction

In recent years, a new type of human-computer interaction has surfaced: gestural interaction. A computer with a gestural interface tries to interpret human gestures via mathemathical algorithms. Gestural interfaces appear in many forms, ranging from a computer that tries to recognize human emotions using a video camera to a gaming console that uses special input devices to detect hand motion (see figure 1.1). Gestural interaction has been popularized recently by mainstream media. For example, a prominent science fiction movie features a gestural interface (see figure 1.2). This fictional interface is controlled directly by hand motions of users.

Figure 1.1: Gestural interaction in a golfing game. *(Image from Nintendo)*

Figure 1.2: Gestural interface in a science fiction movie. *(Image from 20th Century-Fox Film)*

Gestural interaction is also possible on touch screens. Users can interact with a computer by drawing gestures on a touch screen, usually with their fingers or with a stylus. A major broadcasting company used such an interface in the reporting of the American presidential election in 2008 (see figure 1.3).

In the case of the golf game on the game console, it is quite clear to users what gestures they can make. Users play a game of golf and they can swing with the input device to hit the virtual ball. But how does a user control the futuristic interface from the science fiction movie? And what happens if you draw a circle on a display wall? Learning to work with these interfaces can be difficult for novice users. This forms the starting point of our research. We focus on improving gestural interaction on touch screens [1] by rendering previews of possible gestures on the screen when users try to draw a gesture. We named this concept **gesture previewing**.

This thesis starts with an overview of touch screen hardware and interaction methods on touch screens to give the reader a basic knowledge about the context in which our research is conducted. After this we look at gestures in more detail and look at their advantages and disadvantages. To offset the disadvantages of gestural interaction on touch screens the concept of gesture previewing is introduced. At TNO, Groningen we have developed a proof-of-concept framework for integrating gesture previews in existing applications. The implementation of this framework and its integration in an existing application is discussed. We also

[1] The futuristic interface is not available yet.

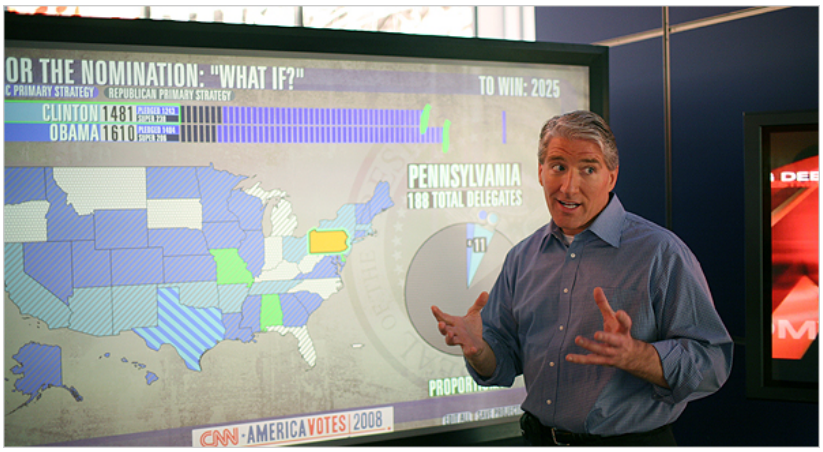

Figure 1.3: Touch screen used in the reporting of the American presidential election in 2008. *(Image from CNN)*

conducted a user study to find out if gesture previews are actually useful and if users like working with them. This user study yielded some interesting results which are also presented. Finally we draw conclusions regarding the usability of gesture previews and suggest areas of interest for future research.

Chapter 2

Touch screen technology

2.1 Introduction

A touch screen is a display that can detect touches in its display area. Users mostly interact on touch screens with their fingers, though sometimes a stylus or other object is used to contact the screen. Touch screens are a relatively new invention. In 1972 a patent was filed for an *Electrical Sensor Of Plane Coordinates* by the university of Kentucky. Strictly speaking this device was not a touch **screen** as it gave no visual feedback. In 1974 a patent was filed describing an *Electrographic Sensor For Determining Planar Coordinates*. This was a transparent sensor pad on top of a display, forming the worlds first touch screen. Since then touch technology has been embraced by mainstream society. Examples of products that use a touch screen for in- and output include ticket machines (see figure 2.1), display walls and game consoles.

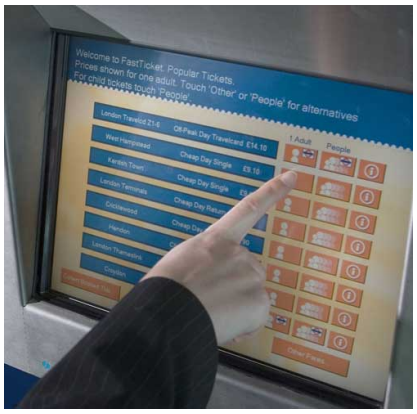

Figure 2.1: A ticket machine with a touch interface

The first generation of touch screens could detect only one touch at a time and could not distinguish the shape and/or pressure of that single touch. But in 1984 Bell Labs engineered a touch screen that could track multiple fingers simultaneously. The last few years multi-touch technology has reached mainstream society and is now integrated in devices ranging from mobile phones to tabletop screens. This chapter gives the reader an overview of the technology behind touch screens.

2.2 Technology overview

2.2.1 Resistive touch screens

A resistive touch screen is composed of several sheets separated by a thin layer of air. In the simplest setup a voltage is applied to one sheet and a device measures the voltage of another sheet. When an object touches the screen the two sheets are pressed together and the voltage of the second sheets changes, indicating a touch. By applying a voltage gradient to the first sheet the contact coordinate of the touch can be deduced. Imagine a sheet to which a voltage gradient is applied uniformly, ranging from 2 Volt on the left side of the sheet to 10 Volt on the right side. If a voltage of 6 Volt is measured in the second sheet one can deduce that the screen was touched exactly in the middle. By combining multiple layers of sheets both the x and y coordinates can be obtained.

Figure 2.2: This gaming device has a resistive touch screen. *(Image from Nintendo)*

Advantages of resistive touch screens are that they are cheap, precise and reliable. However, the screen can be damaged by sharp objects. Another problem with resistive screens is that it is hard to detect multiple touches at once, since

every sheet can only detect a single coordinate. Because of this other technologies are better suited for multi-touch screens. An example of device with a resistive touch screen can be seen in figure 2.2

2.2.2 Capacitive touch screens

A capacitive touch screen makes use of the fact that the human body exhibits capacitance. Capacitive sensors can measure this. By comparing input from multiple capacitive sensors (for example four sensors placed at the corners of a screen) algorithms can determine the position of a touch. A drawback of capacitive touch screens is that they do not work if you touch them with a pen or while wearing gloves because they rely on conduction. An advantage of capacitive touch screens is that they are durable and therefore usable in public places and industrial settings. Also, a sensor field can detect multiple capacitance changes at once, making it possible to detect multiple touches at once as well. Figure 2.3 shows a user playing a game on a device with a capacitive touch screen [1]. Thanks to the capacitive touch screen the user can control the game with multiple fingers at once.

Figure 2.3: A popular mobile device with a capacitive touch screen. *(Image from Apple)*

2.2.3 FTIR

When a ray of light strikes a medium boundary it is partially reflected and partially refracted. This effect can be observed in a glass of water. However, when an incoming light beam strikes a medium boundary at a small angle it will be completely reflected. This effect is called total internal reflection and allows light to travel through a fiber-optic cable.

Total internal reflection can be disturbed by placing a third medium with a high

[1] see http://en.wikipedia.org/wiki/IPhone for more details about this device

refractive index close to the old medium boundary. This effect is called FTIR (frustrated total internal reflection). Jeff Han [7] discovered that it is possible to build touch screens by exploiting this effect. To do so, a plate of acrylic [2] is flooded with infrared light from the sides. The infrared light beams are 'trapped' inside the acrylic due to total internal reflection. When the plate of acrylic is

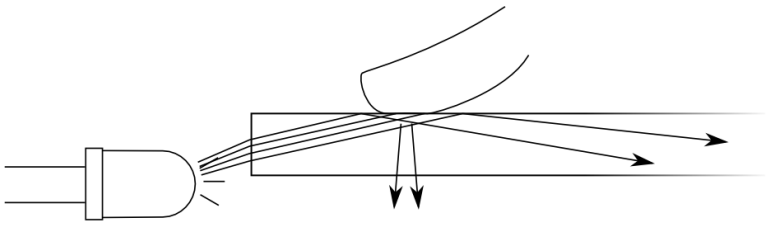

Figure 2.4: Total internal reflection is disturned when a finger touches the screen.

touched by a material with a high refractive index, such as a finger, the internal reflection is disturbed and the infrared light beams will scatter. This effect is demonstrated in figure 2.4. An infrared camera positioned beneath the plate records the scattered light beams and produces footage as shown in figure 2.5. By using image processing techniques touch data can be extracted from this footage in real time. This technique has several advantages. FTIR touch screens can

Figure 2.5: Frustrated reflection as captured by an infrared camera.

detect multiple touches at once and can also provide information about the size and shape of these touches. For example, a pen touch can be distinguished from a finger touch. FTIR touch screens are surprisingly cheap and simple to build. Since acrylic is transparant a standard beamer can be used to project a display

[2] Other materials are used as well.

on the touch surface. On the internet several do-it-yourself guides are available [3] for building FTIR touch screens and open source libraries can convert the camera footage into abstract touch data that can be used by applications [4]. Another advantage of FTIR is that the technology is scalable. The previous two techniques do not work very well on larger touch screens because their accuracy decreases or their price increases as screens get larger.

A disadvantage of FTIR is that processing the video footage is computationally expensive. But with processing power ever increasing current video processors can track multiple points of input at rates of over 30 frames per second. Another problem with FTIR touch surfaces is that the camera has to be placed at some distance beneath the top plate to capture all footage. This makes it impossible to create flat FTIR touch surfaces, for example one that hangs on a wall. All implementation work discussed in this thesis is developed and tested on a FTIR touch screen available at TNO, Groningen. This table can be seen in figure 2.6. Note that the table is massive due to the limitations of the technique.

Figure 2.6: FTIR tabletop touch screen available at TNO, Groningen.

2.2.4 Other technologies

Resistive, capacitive and FTIR touch screens are popular but there are many other ways to construct touch screens. A popular technology worth mentioning is the surface acoustic wave, or SAW, touch screen. In SAW touch screens, ultrasonic sound waves are transmitted over the screen surface. A touch interferes

[3] For example: http://lowres.ch/ftir/
[4] Noteworthy is Touchlib: http://www.nuigroup.com/touchlib/

with the sound waves and this interference is detected by sensors placed around the screen. The exact location of a touch can be deduced from the distortion of the incoming sound waves. A challenge with these types of touch screens is that contaminants or moisture on the screen can interfere with the ultrasonic sound waves. This makes SAW touch screens not very suitable for harsh environments.

A technique that works in a similar way is infrared flooding. An array of infrared LEDs floods the surface of a screen with infrared light beams. Photosensors on the other side of the screen measure the incoming light beams. A touch of the surface will interrupt one or more of the beams, causing a decrease of brightness measured in one or more photosensors. An advantage of infrared flooding is that the sensor grid can detect multiple touches at once. And the infrared overlay can be implemented as a simple frame around a standard screen, requiring no image-quality degrading surface coating.

Another touch screen technology that is becoming increasingly popular is optical imaging. This technique uses cameras placed in the corners of the screen. Footage of multiple cameras can be triangulated to determine the exact location and shape of a touch. Advantages of this method are that it is affordable, scalable and does not require a surface coating.

2.3 Conclusion

In this chapter the hardware part of touch screens has been discussed. A brief knowledge of underlying technology is useful for experts involved in the design of touch screen interfaces. Different types of touch screens can differ in price, accuracy, speed, size, power consumption and the ability to handle multiple touches at once. Developers and interface designers have to take these constraints into account while working on a touch interface.

However, for most end users the underlying technology is not important as long as they can interact with their hardware in a pleasant and effective way. Human-computer interaction on touch screens is discussed in the next chapter.

Chapter 3

Interaction on touch screens

3.1 Introduction

Touch screens give users the possibility to interact with computers in new and different ways. Additional input devices such as keyboards and mice are no longer a necessity. On the other hand, touch screens have their shortcomings. For example, a mouse has multiple buttons and those buttons can have different functions. A user can use the the left mouse button to open a folder and the right mouse button to open a context menu. This is not possible on a touch screen - there is no such thing as a *left touch* on a touch screen. The question arises whether it is possible to create interfaces that take advantage of the new possibilities of touch surfaces and yet disguise their shortcomings.

Interesting research has been conducted to indentify challenges and to find and test new interaction methods for touch surfaces. This chapter presents the reader with an overview. Every section discusses a different challenge for interface designers and presents new interaction techniques that are proposed to tackle that challenge.

3.2 Porting a traditional interface

The simplest way to create a touch screen interface is to make an existing interface compatible with touch screens. Manufacturers of personal computer operating systems have already started to do so [1]. However, porting a traditional interface is not always the best option. As an example, consider the console shown in figure 3.1. If this console is ported to a touch screen, how should users input text? There are solutions, for example handwriting recognition or the usage of

[1] For example, Windows 7 has extensive multi-touch support

an on-screen keyboard, but these solutions suffer from several drawbacks, for example:

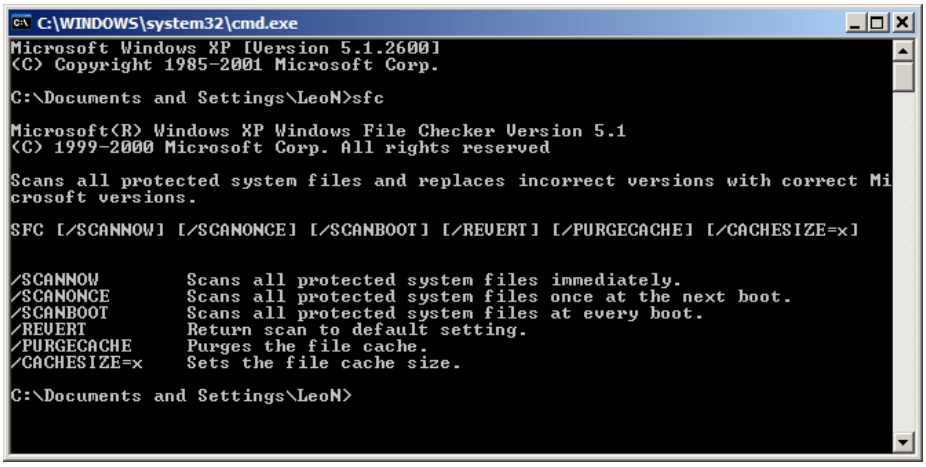

Figure 3.1: An interface that is not very useful on a touch screen. *(Image from Microsoft)*

- Speed of text entry. Using a keyboard, an average typist can reach speeds of up to 40 words per minute. This is much faster than humans can write [1]. Thus, for applications that require extensive text input handwriting recognition is not a suitable option.
- Lack of flexibility. Another problem with handwriting recognition is that a keyboard has lots of keys that are not mapped to a letter, such as function- and modifier keys. These keys are often used to speed up common actions such as saving a document. These tasks can become tedious when using handwriting recognition.
- Screen real-estate. An on-screen keyboard (see figure 3.2) uses a lot of screen space. That makes it difficult to present information to users while they are entering text. This is especially a problem on smaller devices. Another problem is that, while users are entering text, their hands are hovering over the screen, obstructing the view even more.
- User preference. Perhaps an underestimated fact is that users might not **want** a new method of text entry. Research by Hinrichs et al. [10] shows us that users might prefer the possibilites, speed and tactile feedback of a real physical keyboard.

The console example shows us that porting a traditional application in the most straightworward way to work on a touch screen is not an optimal solution in all situations. The lack of a keyboard can be a nuisance for users. In a similar way,

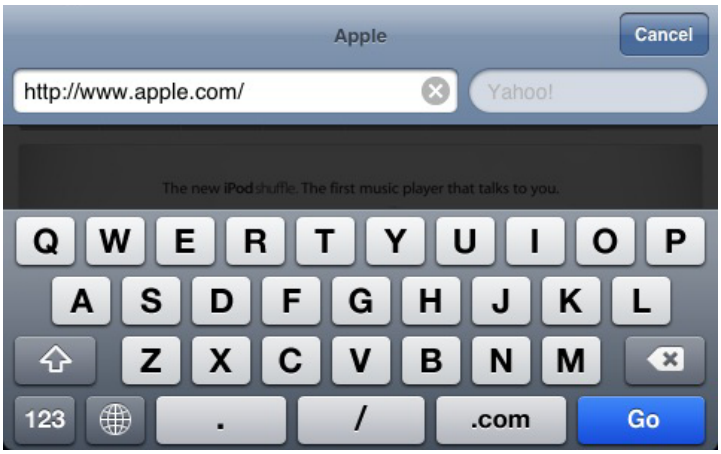

Figure 3.2: On-screen keyboard. *(Image from Apple)*

research by Forlines et al.[5] suggests that mouse input is faster and more precise than direct-touch input. This means that a touch screen application might not be the best solution in every domain. Text editors or first person shooters are maybe a more pleasant experience on traditional computers.

Researchers are trying to solve these issues by inventing other way of porting existing interfaces to touch screens. *DTmouse* [4] is a method of improving the accuracy of touch input and can simulate mouse buttons by tracking multiple fingers. And new, faster methods of text entry on touch screens are currently emerging [2]. In any case, a designer should be aware that he cannot simply port old interfaces and interaction methods to a touch surface and expect them to work flawlessly.

3.3 Novice versus expert users

Another challenge that designers face while creating a touch screen interface has to do with usability. An ideal interface should be easy to understand for new users, but also has to accomodate expert users. An example of this is a menu in a text editor. A novice user can browse through the menu, searching for his desired goal. For expert users this is tedious and therefore all options are also available using keyboard shortcuts. Such an interface accomodates both types of users. An example of an interface that violates this principle is given in figure 3.3. This personal assistant tried to help users in a popular text editor. That can be useful for novice users. However, there was no way to turn off this avatar and expert users found the avatar distracting and annoying.

[2] http://www.swypeinc.com/

Figure 3.3: Personal assistant in a text editor. *(Image from Microsoft)*

Research is ongoing regarding the usability of touch screens for both novice- and expert users. For example, Vanacken et al.[2] have come up with *TouchGhosts*, virtual avatars that appear on touch screens to help users. A screenshot of their interface is given in figure 3.4. This is an interesting approach but the tedious explanations and the avatar that is always visible on the screen lead us to believe that this interface concept can be annoying for expert users, just as the personal assistant discussed previously. The authors do not mention this potential problem in their article.

Figure 3.4: Screenshot of *TouchGhosts* demonstrating how to zoom in on a photo.

3.4 Cooperation and territoriality

Other challenges arises for those who design interfaces for large touch surfaces. These surfaces can support, in theory, input from- and output to multiple users

simultaneously. For example, different stakeholders can cooperate around a tabletop screen in an emergency control room. Or several children can play an educational game together on a tabletop touch screen. How should visual information be oriented on such screens? If one user can read a text, it might be upside-down for a user standing across the table. Also, research by Carpendale et al. [20] shows that users behave territorial on tabletop screens. In other words, they imagine the part of the touch screen next to them is *theirs* and unconsciously do not appreciate other users touch the screen there.

Figure 3.5: A fluid interface: components flow around the border of the screen

New interaction methods have been designed to tackle these problems. Hinrichs et al. [9] came up with the fluid interface (see figure 3.5). In such an interface, components flow over the touch screen. This way the components can successively be used by multiple users without invading the privacy of other users. A drawback of this method is that users sometimes have to wait before they can interact with specific interface components.

Another example of a new touch screen interface concept is *superflick*, a technique that can be used to share interface components by sliding them across the table [17]. However, the concept explained in this article might not be very intuitive for novice users.

To solve the problem of orientation on large touch screens Hancock et al. [8] have compared five different methods of rotating and translating interface com-

ponents. Their conclusion is that no single method is the best solution for all applications, again showing that designing a powerful yet intuitive interface for touch screens is not an easy task.

3.5 Conclusion

In this chapter the problems involved in designing touch screen interfaces have been discussed, as well as several proposed solutions. The reader has seen that porting traditional interfaces to touch screens is not always the best solution. These interfaces suffer from several problems when ported and do not take advantage of new possibilities offered by touch screens.

A particular method of interaction that has not been discussed yet is gestural interaction, a different way of interacting with computers which is especially geared towards touch screens. Users control their computer by drawing *signs* on the screen. Such a gestural interface might not suffer from the drawbacks listed in this chapter. My research focuses on the usability of gestural interfaces. Gestures are discussed in more detail in the next chapter.

Chapter 4

Gestures

4.1 Introduction

In general, gestures are forms of non-verbal communication. They can function as a shortcut, used instead of or in combination with verbal communication. For example, you can nod your head if you agree with something. The meaning of a gesture often depends on the context in which it is used. Extending your index finger and your thumb at the same time can be interpreted as a greeting, but also as pointing a gun at someone. And in a different orientation it represents the letter 'l' meaning 'loser'. In different countries, social groups and contexts the same gesture can have a completely different meaning.

Figure 4.1: A gesture that can be interpreted in multiple ways

The same point also hold for gestures on digital devices. In the context of of touch screens, gestures are figures drawn by users on the screen. As an example, a user could draw a cross on the screen to close an application, instead of browsing through a menu to do the same thing. In this chapter we take a closer look at gestures (focusing on touch screens) and discuss their advantages and disadvantages.

4.2 The definition of a gesture

Gestural interaction is somewhat related to handwriting recognition. Users draw signs on a touch screen and these are subsequently interpreted by the device. But a more formal definition of a gesture is essential if one wants to store and exchange gestures, develop tools and frameworks that can simplify development of gestural interfaces, or support similar gestural interfaces on a wide range of devices. Surprisingly, a widespread standard gesture format does not exist yet.

For our purposes we define a gesture based on the following two properties: First, every gesture can be represented by a set of measured positions in space. Second, these positions are measured over time. The way a gesture is drawn over time can change the meaning of a gesture. This leads us to the following definition:

> A gesture is a set of measured points P in space and a corresponding set of time intervals T between measurements.

No touch screen is mentioned in this definition. Indeed, gestural interaction is not limited to touch screens. Other input devices that do detect motion and thus allow users to perform gestures include mice, digital gloves and camera tracking systems. The research presented in this thesis focuses on touch screens and therefore these other devices will mostly be ignored.

Figure 4.2: Posture based rendering

The given definition encapsulates what a gesture **is** but not what it **does**. As with real-world gestures, their meaning is derived from their context. An interesting

notion is that some input devices can measure more than only movements. A mouse has buttons that can be pressed. A gesture might have a different meaning depending on which mouse button was pressed. And as discussed in chapter two some touch screens can make a distinction between touches of different shapes. Grubert et al. [6] make use of this in a painting program (see figure 4.2) for touch screens in which the posture of your hand influences the drawing style.

The multi-touch table at TNO, Groningen and its accompanying software were only capable of measuring single points of input. Therefore other possible measurements were ignored during our research. This also has the advantage that our research is relevant for a wide range of devices.

4.3 Multi-touch and cooperative gestures

As discussed in chapter two, the latest generation of touch screens can detect multiple points of input at once. These devices are called multi-touch screens. On such devices users can make gestures with multiple points of input. Such a gesture is called a multi-touch gesture. A common example is given in figure 4.3. Users can intuitively zoom in and rotate pictures with two fingers. This is called pinching.

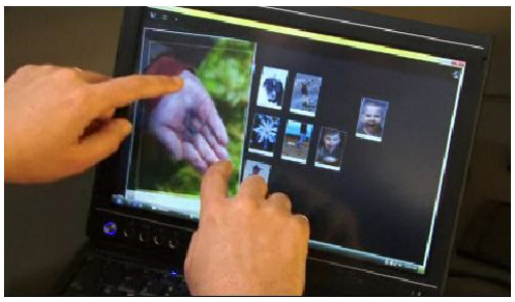

Figure 4.3: A common multi-touch gesture *(Image from Microsoft)*

The term multi-touch gesture does not imply that all points of input are created by a single user. And indeed multi-touch gestures can be drawn by multiple cooperating users. These gestures are a special type of multi-touch gesture and are called cooperative gestures. To prevent confusion between terms we stick to the following definitions:

- Multi-touch gesture - a gesture with multiple points of input drawn by a **single** user.
- Cooperative gesture - a gesture drawn by **multiple** users.

Cooperative gestures are a relatively new interface concept, being introduced by Morris et al. [14] in 2006. To demonstrate the concept they implemented a drawing application in which cooperative gestures featured an important role. An example can be seen in figure 4.4. If all users put their hand palm down on the touch screen for a certain amount of time the drawing area will be cleared. This way a drawing can only be completely erased if all users want to. Such an example leads us to believe that cooperative gestures can be a useful tool in decision-making situations.

TNO Groningen is developing a proof-of-concept emergency control room application [1]. This application runs on a large tabletop display. The police, the fire service and other stakeholders stand around the tabletop and cooperate to manage natural disasters or terrorism threats. In such a setting, decision-making can be supported by allowing all users to draw votes on the screen. Drastic measures, like sending in the riot police, will only be taken if all stakeholders agree and cooperate with the gesture.

Figure 4.4: An example of a cooperative gesture.

Please note that the definition of a gesture given in the previous section does not support multi-touch gestures. This was a conscious decision since our research focuses on single-touch gestures.

4.4 Local and global gestures

One can make a distinction between two different types of gestures. To demonstrate this we introduce two sample mouse gestures. One of the most-used gestures on personal computers is the drag-and-drop, first introduced by Apple on their Macintosh computers. Dragging an icon is performed in the following way:

[1] The application will be discussed in more detail in chapter 6

- 1.Hovering your mouse pointer over an icon.
- 2.Moving the mouse pointer to a desired location for the icon.
- 3.Holding the left mouse button.
- 4.Releasing the left mouse button.

Gestures are also implemented in some browsers. In a particular browser [2] one can go back a page in the browsing history by:

- 1.Holding down the right mouse button.
- 2.Moving the mouse to the left.
- 3.Releasing the right mouse button.

There is an interesting distinction between the purposes of these two gestures. The drag-gesture is local. What it represents depends on where you start the gesture. If your mouse pointer is hovering over an icon the drag-gesture will only move that specific icon. And if your mouse pointer is hovering over an empty part of your desktop nothing will happen at all. This is different from the gesture in the Opera browser. It does not matter where you start this gesture. Neither does it matter where you end the gesture. The gesture is detected whenever you make a sweep to the left with your mouse while your right mouse button is held. The back-gesture is global. Indeed digital gestures can be divided in two groups: local and global gestures. Gestures from these groups of gestures are used for different purposes. Local gestures serve to interact with virtual objects, while global gestures serve as a shortcut for commonly used functionality.

Because of these conflicting goals, local and global gestures are performed in different ways. With the drag-gesture you can place any icon exactly where you want it. But this requires very precise mouse movements and visual feedback. This means that the gesture has to be performed in a slow, controlled manner. On the other hand of the spectrum is the back-gesture. This gesture is usually executed in a swift way, without any feedback. This is logical: the gesture serves as a shortcut. If executing it takes a long time it would be easier for the user of the browser to press the back-button in the user interface instead.

The distinction between local and global gestures is apparent on touch screens too. An example of a local gesture would be to zoom in on a picture (the pinching that was previously discussed). A global gesture can be drawn anywhere and might be used to exit an application or open a menu. This distinction is the cause

[2]Opera - see http://www.opera.com/

of some problems on touch screens. If a user draws a gesture on a picture, what does he want? Is his goal to move the picture around or does he want to exit the picture viewing application? In the previous two examples mouse buttons were used to make a distinction between local and global gestures but this is not possible on touch screens.

4.5 Discussion

Interface designers should consider gestural interfaces for touch screens because such interfaces have several advantages. First of all, gestures are very efficient. Just as keyboard shortcuts speed up common tasks on personal computers, gestures can speed up the interaction with a touch screen. A simple gesture like a circle can easily be drawn within a second. Such a gesture can be coupled to the execution of a common task.

Second, the concept of drawing what you want on the screen is very intuitive and users find it pleasant to do. People draw pictures for fun and to expres their feelings and can easily map this concept to a touch screen.

Figure 4.5: Drawing for fun.

Another advantage of gestures is that they work regardless of orientation. As discussed in the previous chapter orientation of interface components is problematic on large tabletop devices. Text that is easily readable for one user can be upside-down for a user standing on the other side of the table. Gestures do not suffer from this problem: they can be drawn by users in any orientation. And global gestures can not only be drawn in any orientation, but also at any

position. This means that there are no territorial issues. A menu or button on a large touch screen can be awkard to use because it is hard to reach or because the the physical proximity of other users is uncomfortable. With global gestures, users can themselves choose where they want to touch the screen. There still can be territorial issues with local gestures but these exist by definiton.

Finally, collaboration on large touch screens can be supported by gestures. Scott et al.[19] have shown that, for collaborative interaction on tabletop displays, it is important to support interpersonal interaction and to provide shared access to physical and digital objects. This can potentially be achieved with cooperative gestures.

A gestural interface also suffers from some problems. One of the main problems is that gestures have no intrinsic meaning. If an interface uses handwriting recognition all users know the range of characters they can draw. In other words, the sign for the letter 'A' is known by everyone who masters the Latin alphabet. However, there is no universal gesture alphabet. Thus, for new users it is problematic to deduce the range of gestures that is available to them. At TNO, Groningen a tabletop device was bundled with a pack of games to demonstrate its capabilities. In this software, a global gesture could be used at any time to select a different game. This gesture is given in figure 4.6. The problem with this interface was that novice users had absolutely no clue about how to navigate the demonstration software. No feedback was given when a user failed to draw the correct gesture. This ment that it was impossible for novice users to switch to a different game unless another user demonstrated the gesture first. This leads us to conclude that gestural interfaces have a steep learning curve.

Figure 4.6: A gesture without an intrinsic meaning

Another problem with gestures is that they are sometimes drawn accidently. This happens even with cooperative gestures. In figure 4.4 a gesture was shown that was used by Morris et al. [14] to clear a drawing board. One of their findings was that a new drawing was sometimes started accidently when multiple users rested with their hands on the tabletop device. One can also imagine that the gesture given in figure 4.6 is drawn accidently in a game setting. It is hard for software to decide whether a gesture is drawn on purpose or not.

Developers and designers of gestural interfaces face challenges too. Constructing

an intuitive set of gestures is hard and gesture detection is a complex subject. Input data has to be filtered and classified. The developer has to have some knowledge about statistical models. And gestural interfaces are relatively new and lack support in common GUI frameworks. Due to the recent uprising of gestural interfaces this problems will hopefully be solved in the coming years.

4.6 Conclusion

We have seen that gestural interfaces have some promising advantages. On a wide range of devices they can be used as shortcuts to speed up interaction. And on large tabletop touch screens gestures can be useful to support multi-user interfaces. But the lack of feedback and the fact that there is no universal gesture alphabet leads us to believe that a gestural interface can have a steep learning curve. We have come up with a solution to this problem which is introduced in the next chapter.

Chapter 5

Gesture previews

5.1 Introduction

In the previous chapter is shown that a gestural interface can be difficult to master for novice users. One of the reasons for this is that novice users do not know what gestures they can draw. The usability of gestural interfaces would increase if they could somehow help to make new users comfortable with all available gestures. Vanacken et al. [2] have experimented with virtual avatars but it is our view that these avatars can distract and annoy experienced users. We have come up with a new way to increase the usability of gestural interfaces that does potentially not suffer from this problem.

5.2 The concept

The definition of a gesture given in chapter 4 shows us that gestures are composed of both a spatial and a temporal component. This implies that gestures are drawn by users with a certain speed. We believe that the speed with which a gesture is drawn forms an indication of the familiarity of users with the gestural interface. An expert user can draw gestures quickly because he knows what gestures are available in any context. On the other hand, a novice user draws gestures slowly because he is not familiar with the motions he is required to make. A user that is unfamiliar with gestures might just touch the screen and do nothing at all, expecting something to happen. This leads us to conclude that the level of help a user requires depends on the speed with which he interacts with the touch screen.

We have come up with the idea of a overlay that is shown on top of any application. Whenever a user hesitates while drawing a gesture, the overlay shows

the user what gestures are available and what actions are coupled to those gestures. We call this concept **gesture previewing**.

5.3 The overlay

If no user is touching the screen the overlay is invisible. The fact that the overlay does not constantly require screen space makes the concept also feasible for implementation on smaller devices. Whenever a user touches the screen, the overlay gradually fades in. This way new users instantly receive feedback about the available gestures. When a user starts drawing a gesture the overlay will gradually fade out. This means that expert users are not distracted: they start drawing a gesture instantly and thus the overlay has no time time fade in. At any time while a user is drawing a gesture the overlay will gradually appear when hesitation (in other words no or slow motion) is detected.

The overlay shows the user all available gestures. It displays them as paths that start at the point where the user is touching the screen. At the end of each path a label is displayed that conveys the action coupled to that specific gesture. Users can initiate an action by following the path that leads to its label with their finger [1]. By following the path they actually draw the gesture. It is our hope that after a few times users remember the required motion and do not need the overlay anymore to draw the gesture. If users want to learn another gesture they can wait for the overlay to appear again.

Kurtenbach et al. [12] have conducted a user study about a similar concept: marking menus. Marking menus allow users to perform a menu selection either by browsing a radial menu or by making a mark with a stylus in the direction of the desired menu item (see figure 5.1). This concept is related to gesture previews: users can choose between a fast method of interaction and a method which gives more feedback. The following two quotes are taken from the article on marking menu's:

A user's skill with marking menus definitely increases with use. A user begins by using the menu, but, with practice, graduates to making marks. Users reported that marking was relatively error free and empirical data showed marking was substantially faster than using the menu. The data strongly indicates there is an advantage in drawing marks even if menus could pop-up instantaneously.

[1] Or stylus, hand, etc.

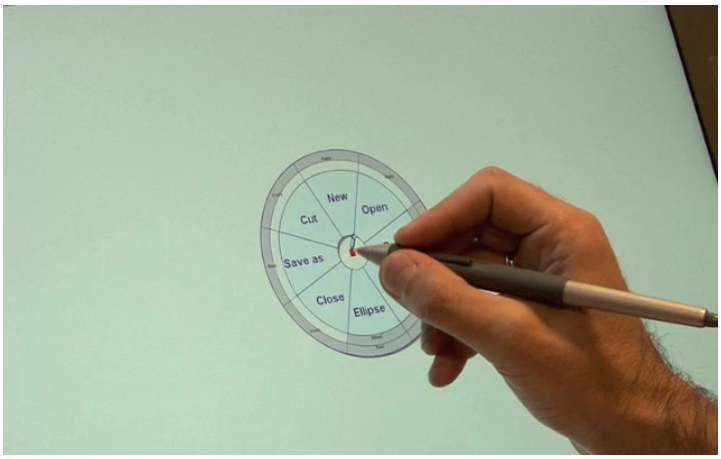

Figure 5.1: Marking menu. *Image from Stephane Huot*

The ability to switch back and forth between menus and marks is important. When a user's skill depreciates during a long lay-off period, the user utilizes the menu to reacquire skills. Our results lend evidence to the notion that for interfaces in general, support for switching between novice and expert mode at the interaction level is a utilized and important feature.

This leads us to believe that our idea has some merits as well.

To make the overlay more versatile, users can also initiate actions by holding their first finger on the screen and touching the label associated with the desired action with another finger. This way, users who have experience with traditional interfaces can use the overlay in a way similar to a traditional menu. A limitation of this method is that it obviously only works on multi-touch screens.

When a user releases his first finger from the screen without drawing a complete gesture and without touching an icon, nothing will happen and the preview overlay will disappear. This way users can intuitively cancel an initiated gesture.

There are various ways to render both paths and labels. A label can be an icon, a text or even a movie clip. A path can be rendered as a single line but also as an animation. There is no universal best solution. On a portable device not much screen space is available so rendering a path as a single line might be the best option. And displaying a movie could be impossible on a portable device because of limited processing power or because it decreases battery life. On a large tabletop device using text labels can be problematic because of orientation issues: a rotating icon is maybe better solution here. Experimentation is required to find a good solution in a specific context.

5.4 Composed gestures

A problem with the overlay described in the previous section is that the screen can become very cluttered if multiple gestures are available. In a text editor gesture previews are surely no option to display all available characters: if a user touches the screen twentysix paths would appear on top of each other, all originating below the finger of the user [2]. This is obviously very confusing. But even in a setting with less available gestures previews can become confusing, especially if their paths intersect. See figure 5.2 for an example of this in another context.

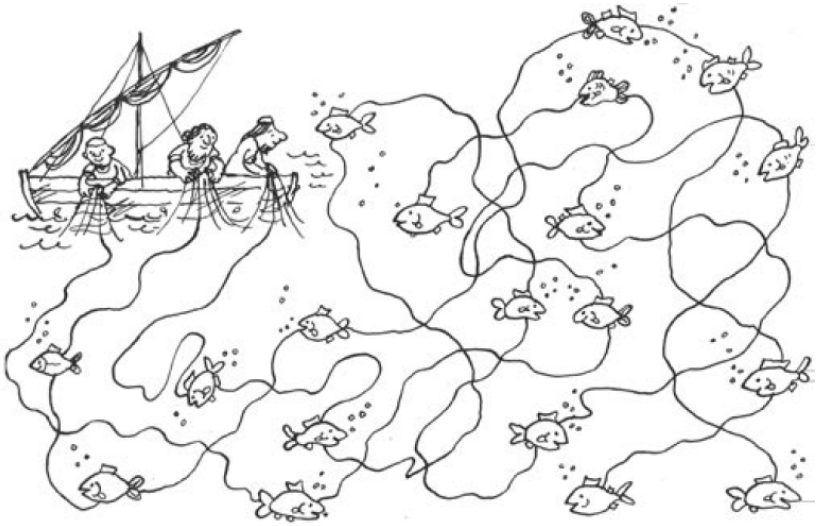

Figure 5.2: Untangling paths can be challenging

To solve this problem we propose a solution where complex gestures are 'composed' from simpler gestures. Imagine a jukebox that uses a touch screen interface. The gestures displayed in figure 5.3 are used to control the type of music the jukebox is playing. The first gesture starts a jazz playlist, the second gesture starts a classical playlist. Both gestures are composed from a leftward curve and a straight line. The overlay can inititally show only the leftward curve with a label indicating 'Change music style'. When a user draws this partial gesture (or when he touches its associated label) a new preview appears that shows a downward line with a label indicating 'Jazz' and an upward line indicating 'Classical'.

In other words, gestures can be represented as a tree. Partial gestures form the edges of the tree and the path to a leaf node represents a complete gesture. Initially the overlay displays the edges available from the root node. When a user draws a partial gesture (or touches its label) he selects an edge in the tree. This

[2] And more if you include capital letters, numbers and special characters.

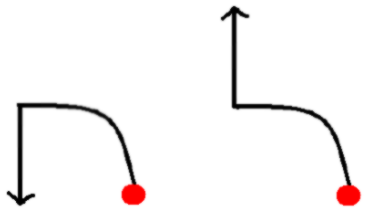

Figure 5.3: Two gestures that start in a similar way.

edge leads to new node in the tree. The overlay can now display a new set of partial gestures that is associated with this node. This goes on until a leaf node is reached, at which point the gesture is finished. To clarify this an example of the gesture tree for changing music styles is given in figure 5.4.

For composed gestures to be useful, a set of gestures should be designed in such a way that similar actions are coupled to gestures that start in an identical way. An example is a set of gestures for editing a document. These gestures can be composed from a partial gesture with a label 'Edit' and three follow-up gestures labelled 'Copy', 'Paste' and 'Cut'. Gestures can also be composed from more than two partial gestures. In the jukebox example the classical music gesture could be followed up by another partial gesture for selecting either Beethoven or Bach. A drawback of large composed gestures is they can become too complicated. It is up to the designer of an interface to group gestures in a meaningful way.

Composing gestures has several advantages. One of them is that the preview overlay can be used in conjuction with a larger set of gestures. In the jukebox example a gesture for playing 'Pop' music can be added. This gesture should be a composed gesture with the same initial part (i.e. a leftward curve) as 'Jazz' and 'Classical'. Now the initial overlay does not grow more cluttered, because initially the gesture for 'Pop' is hidden, just as the gestures for other music styles. Another advantage of composed gestures is that users can memorize partial gestures instead of complete ones. If a user remembers the partial gesture 'Change music style' he can draw it immediately to see what follow-ups are available, skipping the overlay for the initial set of partial gestures. At any moment while drawing a gesture users can receive relevant feedback based upon the partial gestures they have drawn so far.

Note that composed gestures are not always usable. They only work if the available actions can be grouped in categories (like 'Edit' or 'Change music style'). It does not make sense to group unrelated actions behind a partial gesture. If the initial set of gestures is large and cannot be categorized it might be better to display different previews over time, i.e. as a slideshow. By starting with showing

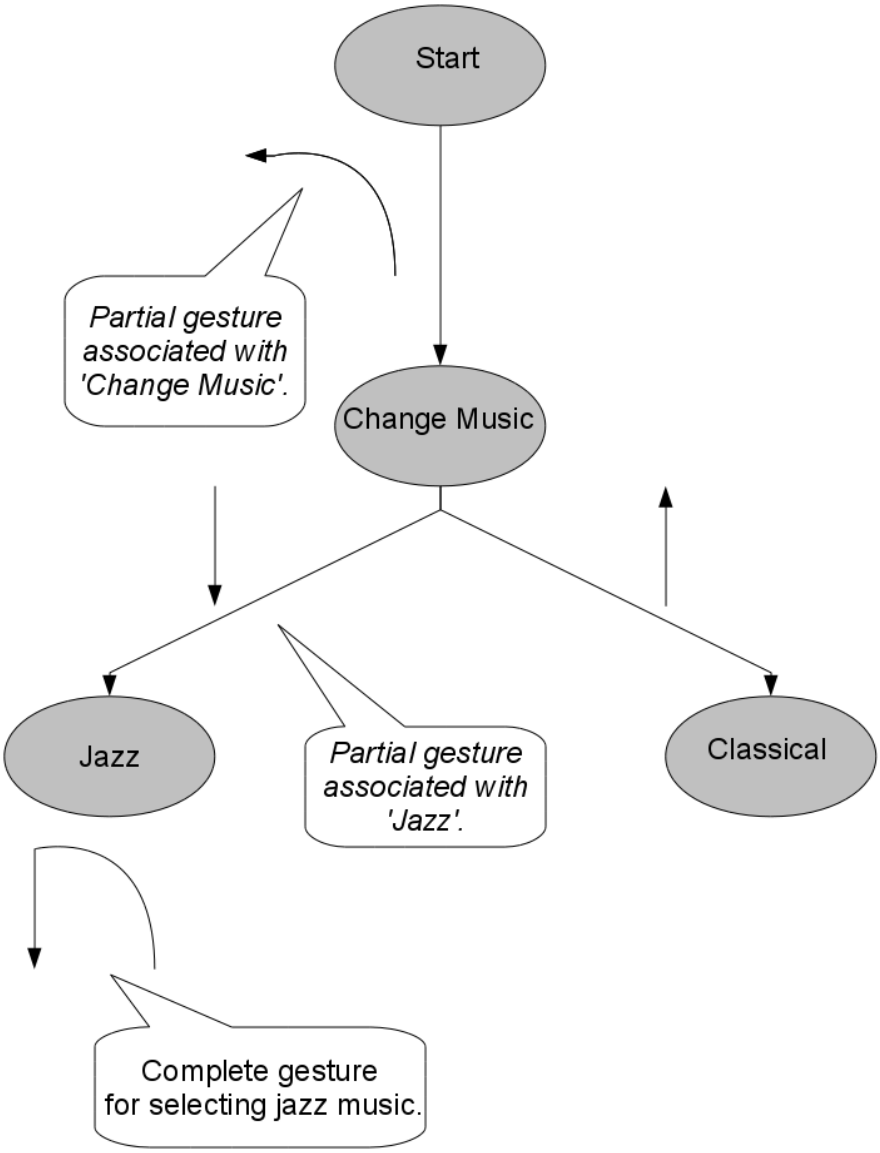

Figure 5.4: An example tree of partial gestures.

the most common gestures and ending with rare ones we keep the average time a user has to wait low. But if possible we prefer an overlay that makes use of composed gestures. In such an overlay users never have to wait for a specific preview to appear.

5.5 Multi-touch previews

Gesture previews can also be used to enhance multi-touch gestural interfaces. When a user touches the screen with his first finger the overlay can show where other fingers can join in to make a multi-touch gesture. And in multi-user settings the overlay can be used by expert users to demonstrate novice users the capabilities of the interface. Even more interesting is that the overlay can in theory support cooperative gestures. The voting scenario discussed in chapter four could be enhanced by gesture previews: if a single user starts a voting gesture the preview overlay can show other users where and how they can join in on the gesture. Gesture previews could also be used in cooperative games to support advanced collaboration between players.

However, in our research we focus on single-touch, single-user gestures. Exploring the possibilies of multi-touch gesture previews is an interesting subject for future research.

5.6 Conclusion

In this chapter we have discussed gesture previewing and composed gestures. A paper about these ideas was accepted for the CHI 2009 multitouch and surface computing workshop. Kurtenbach et al. [12] have reached favorable conclusions about marking menus, a concept that is somewhat related to gesture previews. This leads us to believe that our ideas have some merit, at least in theory. The next step is to find out whether gesture previews also work in practice.

Chapter 6

Implementation

6.1 Introduction

At TNO, Groningen we have developed a framework called **GesturePreviews** that adds support for gestures and gesture previews to existing applications. Developers provide the framework with a list of gestures, after which the framework takes care of recognizing gestures and rendering an overlay with gesture previews. To test the framework it was integrated with an emergency control room application that is currently in development at TNO. A screenshot of this application can be seen in figure 6.1.

Figure 6.1: Emergency control room software. *Image from of TNO*

In this chapter the implementation and integration of our framework is discussed, starting with the design choices we faced during the initial phase of our project.

6.2 Design Choices

6.2.1 Gesture Storage

Adding gestural interaction to applications should be as easy as possible. A good solution for this seems to be an IDE (integrated development environment) that supports creating and editing gestures and that can include these gestures in software projects. But implementing such an environment requires a lot of resources and falls outside the scope of our research. However, to support such environments in the future we can define a standard way of storing gestures. A standardized gesture file format can also encourage the sharing and re-use of gestures.

A file containing several gestures can be created by an interface designer. On application startup, this file is loaded by our **GesturePreviews** framework. The contained gestures are used by our framework to generate gesture previews and to match user input. Application developers do not have to bother with recognizing gestures and rendering gesture previews themselves. This keeps the architectural design of applications loosely coupled. A file format for storing and editing gestures has to meet several requirements:

- The format should be platform-independent. This way gestures can be re-used on a wide range of devices.
- The format has to support gesture previews, i.e. it has to support composed gestures and labels.
- The format should be extendable. Third parties might want to add support for additional features, for example audible feedback.
- Ideally the format is text-based. This way gestures can be created and edited even when no tools for this purpose exist yet.

To meet these requirements we have designed an XML-format for storing gestures. XML is platform-independent and text-based, meeting two of our requirements. Our gesture file format supports versioning: the format is designed in such a way that adapting it in later stages is as easy as possible. Our framework ignores any elements it does not understand so third parties can extend our format with their own XML-elements. And finally, the format supports composed gestures.

A partial gesture is stored in a **segmentDeclaration** element:

```
<segmentDeclaration name="Line" uniformScaling="false">
<point x="0" y="0" />
<point x="0" y="0,1" />
```

```
</segmentDeclaration>
```

This segment is called 'Line' and is composed of two points, $(0, 0)$ and $(0, 0.1)$. In other words this segment is a line oriented upwards. *UniformScaling* is a property that controls whether the segment should be scaled on screens with different aspect ratios [1].

```
<gesture name="Edit">
<segment name="CurveLeft" angle="0"/>
<segment name="Line" angle="0"/>
<preview source="scissors.png"
   relativeWidth="0,05" relativeHeight="0,05" />
</gesture>
```

Several segments together form a **gesture**, which represents a complete gesture. The element **preview** contains information about the label that is displayed in the gesture preview overlay to indicate the meaning of a gesture. In this example, the gesture named 'Edit' has an associated icon that is probably a pair of scissors. The *angle* property controls the angle with which segments are interconnected to form a complete gesture. In this case the angle is 0, all segments are attached to each other without rotation.

Designers can create a set of segments and complete gestures in a single XML-file. An example of such a file is given in appendix A. Our framework can load these files and constructs a tree of partial gestures as described in the previous chapter. This tree is then used to draw correct previews at any time based on user input so far. Note that these XML-files do not contain information about what a gesture **does**. As discussed in chapter three, the exact meaning of a gesture depends on its context. Therefore, application developers have to couple the recognition of a gesture to an action themselves.

6.2.2 Gesture detection algorithm

A topic that is not discussed so far is gesture recognition. When users interact with touch screens their gestures are recorded as a set of points measured over time. We need an algorithm that can classify this set of points, i.e. compare it with an arbitrary number of gesture templates and select the best match. Designing such an algorithm is difficult because user input is sloppy. When a user draws a circle it is always an approximation, meaning that the classifier can only identify a circle within a certain margin of error.

[1] Otherwise, gesture previews can appear 'stretched' on devices such as widescreen televisions.

Researchers have proposed countless solutions for gesture recognition. Possible approaches are to use Hidden Markov Models [13], neural networks[16] or statistical classifiers[18]. A recognizer that is to be used in our framework has to meet several requirements.

- The recognizer should be reasonably easy to implement in our framework. We focus on gesture previews and do not want to spend too much time on making the recognizer work.
- The recognizer should work regardless of orientation. We test our framework on a tabletop device and users will draw gestures in different orientations.
- The recognizer has to be fast. To display gesture previews we require a recognizer that can match partial gestures on the fly.

The best case scenario would be to integrate an existing software library that does gesture recognition in our framework. However, no available recognition library suits our needs. This means we have to implement a gesture recognizer ourselves. We do not want to implement any of the approaches given before. These solutions are hard to implement and require knowledge of statistical and neural models that we do not have. Therefore we have selected the $1 Gesture Recognizer algorithim proposed by Wobbrock et al.[21]. In their article they describe the $1 recognizer algorithm as:

[..] a simple recognizer that is easy, cheap, and usable almost anywhere. Despite its simplicity, it provides optional rotation, scale and position invariance. [..]

An examination of the algorithm showed that the $1 algorithm is indeed relatively simple to implement and is rotation-invariant, meeting our first two requirements. Wobbrock et al. do not mention enough on performance for us to know for sure that their algorithm is fast enough for us. But given that other recognizers did certainly not meet all our requirements, we decided to stick to the $1 recognizer. Luckily it proved to perform adequately.

6.2.3 Rotation and orientation

The gesture recognizer we have selected is rotation-invariant, meaning that gestures are detected regardless of the orientation in which they are drawn. Unfortunately, orientation is still an issue for the gesture preview overlay. Gesture previews should always face the user. If they are oriented otherwise, users will have problems recognizing icons and reading text. This can be problematic on tabletop devices. In this setting the screen can be touched by users standing at

arbitraty positions around the tabletop. Most tabletop touch screens, including the device available at TNO, Groningen, have no way to detect where users are situated. In fact, most touch screens cannot even detect *which* user is touching the screen. This means that it is impossible to orient gesture previews towards the user at all times.

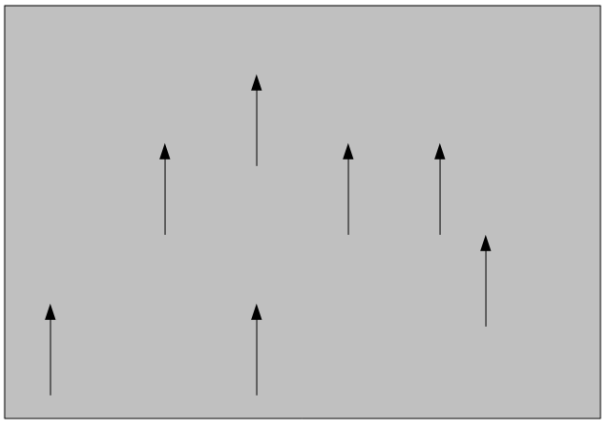

Figure 6.2: Orientation of gesture previews in classic view. Suitable for most devices.

To solve this problem, the **GesturePreviews** framework features two different modes of orienting gesture previews. The first one is 'classic mode' (see figure 6.2) and is the default setting. In this mode, gesture previews are always rendered in the same orientation. This works on portable devices and wall screens because users always face these devices in the same way, just as on a classic computer monitor interface components are always oriented upwards. On a tabletop device this mode might cause problems. In this case the touch screen is oriented horizontally meaning that previews always are oriented towards one side of the table, making them difficult to use for users standing on other sides of the table.

For tabletop devices we have designed another setting called 'center mode' (see figure 6.3). In this mode, gestures are aligned from the point of contact towards the center of the screen. In other words, if a user touches a tabletop device in the upper-right corner the gesture preview is oriented from the upper-right corner to the center of the table. The idea behind this is that users behave territorial (see chapter three) and therefore are most likely to touch the screen closely to where they are standing. And if users do so gesture previews are always oriented correctly in 'center mode'. But a disadvantage of this mode is that gesture previews are rendered in a wrong orientation if a user touches the screen at an area other than directly in front of him.

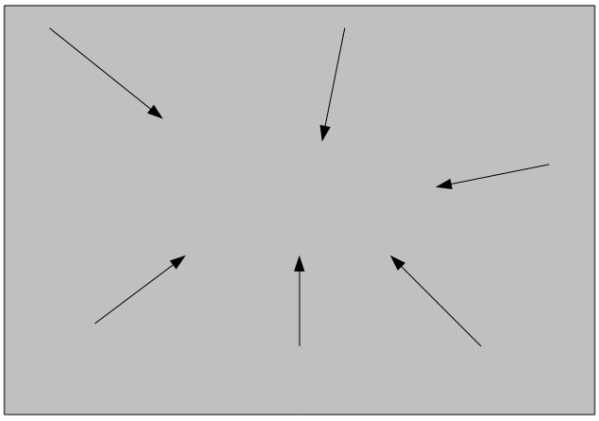

Figure 6.3: Orientation of gesture previews in center view. Useful when multiple users stand are working on a tabletop device.

Orienting gesture previews in the right way is challenging. Our hope is that in the future better hardware can detect the orientation of users to solve this problem completely. For now we have implemented two different solutions in our framework but both have their disadvantages.

6.3 Framework design

The **GesturePreviews** framework is implemented in C# and all rendering is done with WPF [2]. We have chosen this platform because other touch screens applications in development at TNO, Groningen are developed using the same platform. Since we want to integrate our framework in these applications using the same platform seems logical. A downside of this decision is that our work is not easily portable to other platforms.

The framework is targeted specifically at FTIR touch screens (see chapter two). To interpret the video footage recorded by the camera in the tabletop we use the freely available **Touchlib** library. This library processes FTIR video footage in real time on the GPU and produces a stream of touch events. These events are consequently processed by our framework. All events are formatted in the open source TUIO protocol [11]. Porting our framework to other devices is trivial if these devices also support the TUIO protocol. Included with the **Touchlib** library is a configuration utility that can be used to finetune image processing. The image processing depends on several external factors such as the brightness in the room where the tabletop is located or smears on the touch screen surface. Using this application (see figure 6.4) we can easily adapt to such circumstances.

[2] Windows Presentation Foundation, see http://www.microsoft.com/net/WindowsPresentationFoundation.aspx

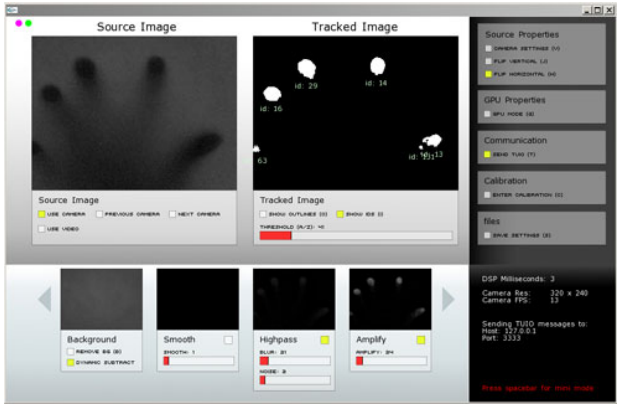

Figure 6.4: Screenshot of Touchlib. *Image from NUI group*

The **GesturePreviews** framework is divided in three main components. A graphics subsystem that generates gesture previews and renders them on the screen. Another component that supports the loading and saving of gestures in our XML-format and constructs a tree of composed gestures from loaded files. And finally a detection subsystem that looks at user input to determine what gesture a user is drawing and what gesture previews have to be rendered at any time.

The architecture of the framework is loosely coupled which means that it is relatively easy to replace one of the subsystems. To demonstrate this we have developed an OpenGL rendering system that can be used instead of the default WPF component. Figure 6.5 shows a screenshot of a game that uses OpenGL for graphics and uses the **GesturePreviews** framework to detect gestures. Gesture previews are drawn in the OpenGL context. The game is still in early development.

6.4 Proof-of-concept application

The emergency control room application currently in development by TNO, Groningen supports emergency services by creating situational awareness. It is designed to run on tabletop devices. The main view is a map (see figure 6.6). On top of this map real-time data is projected, for example the location of fire trucks, photos taken by a local police unit or sensor data from weather monitoring units. All this incoming data can assist managers of emergency services with decision-making. The setup given in figure 6.6 is designed for two stakeholders. The top of the screen is filled by a red toolbar. This side of the table is meant

Figure 6.5: Screenshot of an OpenGL implementation of the rendering subsystem.

to be used by the fire department. The opposite side of the table is reserved for the police force. The blue toolbar contains police-specific actions, while the red toolbar is tailored for the fire department.

The emergency control room application supports two-way communication with mobile clients and VOIP telephony. Not only can managers make a decision based on incoming data, they can also convey these decisions to units 'in the field'. For example, the location of police cars is displayed in real-time on the map. If incoming webcam footage shows an accident has happened, a police chef can select the closest police car on the map, call it and forward it a picture of the accident, all using the tabletop interface.

We integrated the **GesturePreviews** framework in the emergency control room application. First, we identified three actions that could be implemented as gestures: zooming in on a specific location, changing the map view and controlling video footage from an external webcam. We then designed six specific gestures to be implented in the emergency control room:

1. Change the map type to 'street view'.
2. Change the map view to 'satellite view'.
3. Zoom in on Boston, USA.
4. Zoom in on Groningen, the Netherlands.
5. Point a webcam to an office space

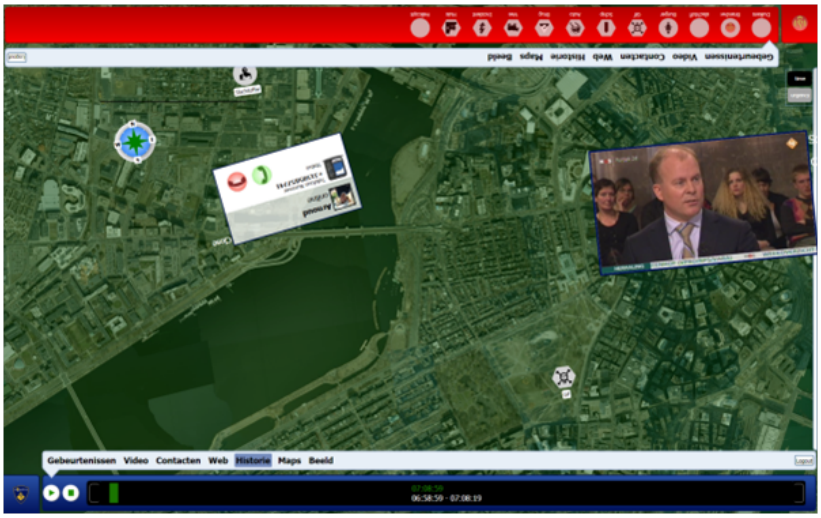

Figure 6.6: Emergency control room application, showing real-time video footage *Image from of TNO*

6. Point a webcam to a parking lot.

The paths associated with these gestures are given in appendix B. Gestures 1 and 2 are related, as are gestures 3, 4 and gesture 5, 6. Therefore we have designed these gestures as composed gestures with the same initial partial gesture. The initial previews overlay shows three partial gestures. One for 'change map type', one for 'zoom to location' and one for 'control webcam'. When a user touches the screen, the overlay becomes fades in until it becomes completely visible. This is shown in figure 6.7. The users sees the three partial gestures that are initially available. Icons indicate their meanings. For example, the webcam icon that shows users that drawing a rightward curve starts a gesture preview.

When a user starts drawing a gesture the overlay starts fading out. This can be seen in figure 6.8. In this case, the user is drawing the partial gesture 'control webcam'. The green tail shows the part of the gesture that the user has drawn so far. When a user has completed a partial gesture he can wait for a new set of gesture previews to appear. This is shown in figure 6.9. The user has drawn the partial gesture 'control webcam' and new previews appear for all composed gestures that start with this partial gesture. In this case, the user can make an upward gesture to zoom in on a parking lot, or a downward gesture to zoom in on an office space [3]. The user can now finish the gesture.

Note that at all stages the user can cancel the gesture by lifting his finger from the touch screen. And instead of drawing the gestures, users can use the gesture

[3]The associated labels are unfortunately not clearly visible in the screenshot.

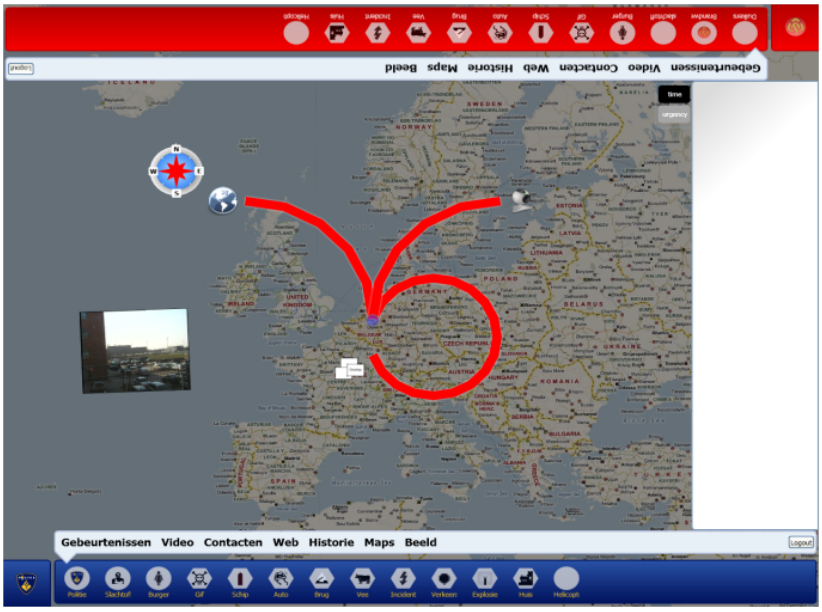

Figure 6.7: Initial overlay with previews. *Image from TNO*

previews as a menu structure and navigate it by touching the labels instead. Expert users who do not require gesture previews can draw a complete gesture at once, as shown in figure 6.10. This also controls the webcam but since the gesture is drawn instantly the overlay will never appear. Users who do know the partial gesture for 'control webcam' but do not know what further actions are available can skip the step displayed in figure 6.7. They can draw a rightward curve and arrive directly at the situation shown in figure 6.9.

6.5 Gesture modes

The six gestures we implemented using the **GesturePreviews** framework are global gestures. These gestures can be drawn anywhere on the screen. This is problematic because the emergency control room application also uses local gestures to navigate the map in the interface. If a user is moving his finger over the map area the application needs to know whether he wants to navigate the map or is drawing a global gesture.

We have come up with a solution to this problem. The emergency control room applications can detect gestures in two modes. In local mode, all gestures are interpreted as local gestures. In other words, users can navigate the map but it is not possible to draw global gestures. In global mode, all gestures are inter-

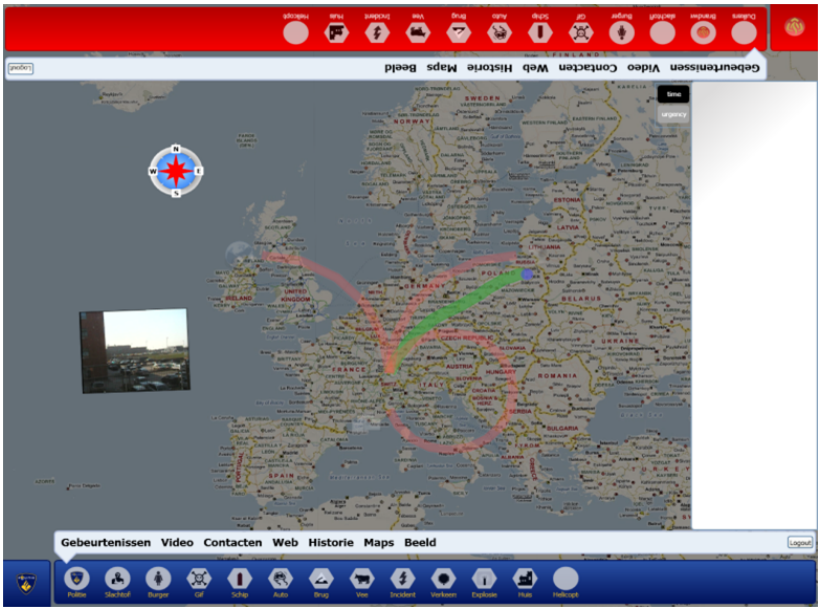

Figure 6.8: The overlay fades out when a user is drawing a gesture. *Image from TNO*

preted as global gestures. This means that users can no longer interact with the map. All gestures drawn on the map are interpreted as global gestures. Users can switch between these modes in two ways:

1. The compass seen in the interface can be touched to toggle between local- and global gesture mode. If the compass is red the application is in global gesture mode. Users can now draw the six gestures introduced previously everywhere on the screen. This has the additional advantage that users can lock the map with the compass, if it is zoomed in on a meaningful location.

2. If a user touches the screen for a second without moving, that touch is considered a global gesture. An explanation will clarify this. If a user is navigating the map, he touches the screen for a second without moving his finger to initiate a global gesture. Then the map is temporarily locked. Gesture previews fade in and the user can draw a global gesture. As soon as the user releases his finger the application swings back into local gesture mode, allowing the user to navigate the map again.

Using this approach users can draw both local and global gestures. A drawback is that this method probably reduces usability for novice users because drawing a global gesture becomes more complicated. This problem is not easily solved. Our application fills the complete screen with a large interface component (the map). If one can interact with that component using local gestures, there is no 'empty'

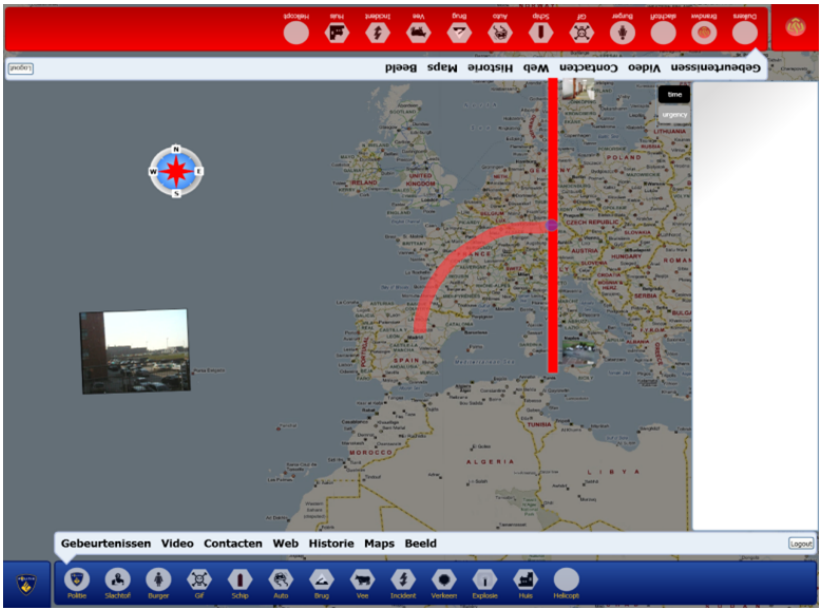

Figure 6.9: A new set of gesturepreviews. *Image from TNO, Groningen*

screen space for users to draw global gestures [4]. Further research to come up with a better way to distinguish local gestures from global ones in such a setting is desirable.

[4] As a counterexample, in a traditional desktop setting there is empty space in the form of the desktop background. A gesture recognizer can interpret gestures drawn on the desktop background as global gestures, while interpreting gestures drawn on other interface components as local gestures.

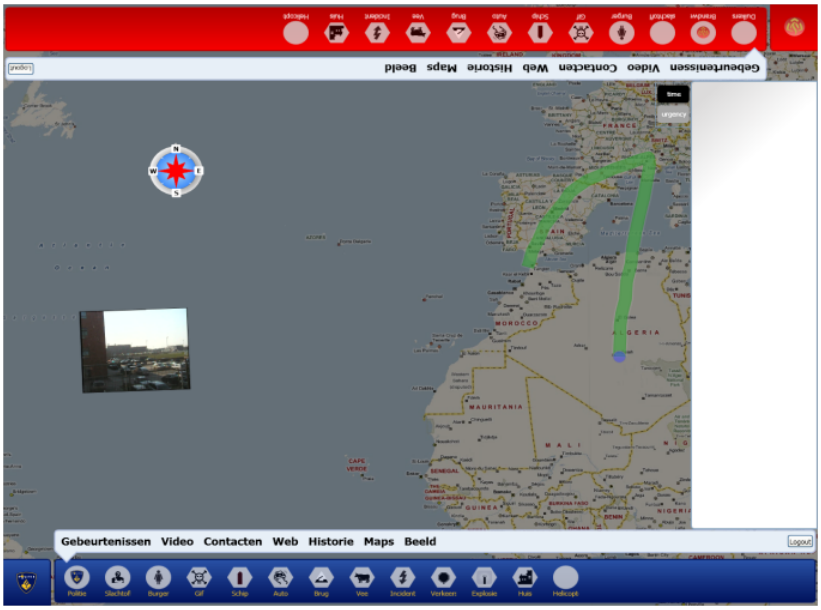

Figure 6.10: No overlay appears when a complete gesture is drawn at once. *Image from TNO*

6.6 Conclusion

In this chapter we have presented our work on the **GesturePreviews** framework. We have elaborated on design decisions and discussed the integration of the framework in an emergency control room application. Using this application we conducted a user study to see if gesture previews are actually useful and whether users like working with them. This study is discussed in the next chapter.

Chapter 7

User study

7.1 Introduction

The concept of gesture previews was designed to meet two goals. First, gesture previews should make gestural interfaces easier to use for new users and second, gesture previews should not be distracting for expert users. To examine whether gesture previews meet these criteria we conducted a user study, using the emergency control room application discussed in the previous chapter.

While evaluating our development work we discovered another potential advantage of gesture previews. Gesture previews show users exactly how to draw a gesture. This means that they can potentially increase the accuracy with which users draw gestures. This would be beneficial as the accuracy of a drawn gesture effects how well a gesture recognizer can classify it. Therefore we decided that the accuracy with which gestures are drawn forms a third criterium to measure the usefulness of gesture previews.

To find out whether gesture previews meet these three criteria we formulated three questions that we need to answer:

- Do gesture previews make a gestural interface easier to use for novice users?
- Do expert users prefer an gestural interface with or without previews?
- Do gesture previews increase the accuracy with which users draw gestures?

To find answers to these questions we designed a user study. This user study was split into two parts. First, users were introduced to the emergency control room application. This application was running on a tabletop touch screen at TNO, Groningen. Users were given the opportunity to experiment with the application

for fifteen minutes. These sessions were recorded on camera and all user input was stored by the **GesturePreviews** framework. The recorded user input was to be analysed afterwards to determine whether gesture previews increase the accuracy with which gestures are drawn, thus answering our third question.

After the session with the emergency control room application, participants were required to complete a questionnaire. Analysis of the recorded video footage, combined with the results of the questionnaire, had to lead to answers to the first two questions. The setup of our user study, its results and the subsequent analysis of these results are discussed in this chapter.

7.2 Users study setup

Our test group consisted of eight users. This group was randomly split into two groups of four users. Half of the users were given a handout with available gestures (see appendix B). For this group, gesture previews were disabled. The other four users were given no information about available gestures. For them gesture previews were enabled. The goal of this setup was to find out whether the second group managed to discover available gestures using gesture previews. The first group served as a control group.

After ten minutes we enabled gesture previews for users from the control group and disabled them for the other four participants. The users now had five more minutes to experiment with the application. This way all participants interacted with the emergency control room application both with- and without gesture previews. During the experiment, we encouraged participants to share their thoughts with us.

7.3 Data capturing and recording

All sessions were filmed with a high-definition video camera. Users signed a document to permit the usage of this footage for scientific purposes. Users stay anonymous as the camera was only recording the surface of the tabletop screen (see figure 7.1).

As part of the user study, all participants had to complete the questionnaire given in appendix C. Note that the questionnaire is in Dutch, as this is the native language of the authors of the user study and of all participants. The first part of the questionnaire had to be filled in before the video session. The goal of this part was to gauge whether participants had experience with gestural interfaces and tabletop touch screens.

Figure 7.1: Video footage of user study. Users stay anonymous.

After the session with the emergency control room application participants had to complete the questionnaire. The second part was focused on the usability of gestural interfaces and gesture previews. After completing the questionnaire participants were given the opportunity to comment freely on any aspect of the user study. Everything was written down and/or recorded for later analysis.

All drawn gestures were stored in a database. This database now contains 758 records in the following format:

```
showpreviews True
normalgesture
time 2-4-2009 11:12:31
match gcircleright
score 5,579876
msduration 3640,625
625,489 -645,2332
630,5363 -630,7058
637,9428 -612,5687
645,0201 -601,7392
[..]
```

The first field, **showpreviews** indicates whether gestures previews were enabled when this gesture was drawn. In this case they were. The second field indicates that this is a 'drawn' gesture, as opposed to a gesture that was entered by touching labels, like this one:

```
clickgesture
time 2-4-2009 11:20:17
match grightcornerup
msduration 4781,25
```

The **time** and **msduration** field indicate at what time the gesture was drawn, and how long it took to draw it. The **match** field lists the template gesture to which the user input was matched. And finally, the **score** field represents the accuracy with which the gesture was recognized (a lower score representing a closer match). This field is only available for drawn gestures, as accuracy is not relevant for gesture that were entered by touching labels. A drawn gesture is also accompanied by a list of all coordinates measured by the **Touchlib** finger tracking library.

7.4 Analysis of video footage and questionnaire

7.4.1 Gestural interfaces

None of the participants had significant experience with gestural interfaces. Participants were asked in the questionnaire to rate their experience with gestural interfaces. Their responses are given below:

Please rate your experience with gestural interfaces.	
I have little experience	3
I have no experience	5

One user mentioned that he used gestures in a web browser. Two users sometimes used gestures on their mobile phone. Since the question given above was too vague we do not know the extent of this experience.

The majority of users had no experience with gestural interfaces at all. But after the user study almost all users were excited about the gestural interface in our emergency control room application:

I enjoy working with this gestural interface.	
Agree Completely	2
Agree	3
Neutral	2
Disagree	1

There is no correlation between prior experience with gestures and these results,s, i.e. users with previous experience are not more (or less) likely to enjoy the ges-

tural interface we presented. In the captured video footage we see users saying they like gestures because 'it feels very intuitive' or 'I can draw them quickly'.

7.4.2 Gesture Previews

Analysis of the video sessions shows that participants like the concept of gesture previews. One participant described it as 'a cool interface'. Afterwards, all participants understood the concept of gesture previews and the majority preferred gesture previews above the paper handout given in appendix B to learn what gestures are available. This is also reflected in the questionnaire:

I rather have gesture previews than a handout with available gestures.	
Agree Completely	2
Agree	4
Disagree	2

However, the authors sometimes interfered with the user study by explaining concepts and by drawing example gestures. It remains to be seen if gesture previews are also intuitive for new users if no expert is around to help them. Our participants are a bit sceptical about this:

Gesture previews increase the usability of gestural interfaces for new users.	
Agree	4
Neutral	3
Disagree	1

Half of the participants doubt that gesture previews make gestural interfaces easier to use for novices. This can partially be explained by the fact that our proof-of-concept implementation suffered from two usability issues. First, the tabletop device available at TNO, Groningen, on which the user study was conducted, was a beta model. The screen was covered by a plastic foil and users had to touch the screen with some pressure or their touches were not correctly detected. But users tend to draw gestures by softly touching the screen with their finger. At several occasions our application did not recognize gestures as such because the user did not press the screen hard enough. This was very confusing and frustrating for the participants, as they have no clue what was going on.

The other usability issue concerned our implementation. As explained in the previous chapter, the emergency control room application can run in two modes. In one mode users can interact with the map using local gestures, in the other mode the map is locked and users can draw global gestures. Users can switch

between these two modes by touching the on-screen compass. Or, in local mode, users can touch the screen for a second without moving to initiate a global gesture. This interface feature was not obvious for the participants of our user study. As a result, users stayed mostly in local gesture mode and could not figure out how to draw global gestures.

Almost all users are positive about another aspect of gesture previews. During the video sessions, several participants say that gesture previews are particularly useful for advanced users, to see what gestures are available in a new context (for example another application).

7.4.3 Gesture drawing speed

In the video footage we see that users draw gestures *slower* with gesture previews enabled. When users are given a paper with available gestures, they draw them on the screen using quick flicks of their fingers. But when gesture previews are enabled users tend to exactly follow the paths displayed in the **GesturePreviews** overlay, sometimes taking up to ten seconds to draw a simple gesture. Participants can be observed saying 'I think like I have to follow the path exactly with my finger' and 'I feel like my finger motions are limited by the gesture preview'.

This change in speed occured instantly when gesture previews were enabled or disabled. In other words, a user that was drawing gesture quickly without gesture previews will follow gesture paths slowly and precisely as soon as gesture previews are enabled. And when gesture previews are disabled again he will revert to his old style of quickly drawing gestures. This behavior was observed by all participants. This leads us to believe that gesture previews can potentially increase the accuracy with which gestures are drawn. But the large decrease in interaction speed is an unwanted and completely unexpected side-effect.

A major cause of this behavior is that users subconsciously wait until the gesture previews overlay has completely faded in. Indeed, almost all participants mentioned that the overlay should fade in quicker, especially if the application is in local gesture mode. In this mode users have to wait almost two seconds [1] before gesture previews are completely visible. This is too long.

7.4.4 Composed Gestures

The emergency control room application used in our user study features six global gestures. These are composed gestures, split into three pairs of gestures that start

[1] One second for locking the screen to initiate a global gesture and subsequently another second for fading in the overlay.

with the same partial gesture. We analyzed the video footage to see whether users understand the concept of composed gestures. All users commented afterwards that they understood the concept. The concept of a preview overlay that adapts to what the user has drawn so far is considered 'an impressive feature' and 'it looks cool'.

But initially most users have some troubles to grasp the concept of composed gestures. We observed for all participants that the first time they draw a partial gesture they release their finger from the screen, thinking they have finished the gesture. Only after experimenting for a few minutes they understand that they have to wait for the next set of previews to appear. Users indicated afterwards that the reason for this is that 'the new paths fade in much too slow' or 'it feels like there is a gap between the first part of a gesture and the second'. This seems to be a usability issue of our implementation. The gesture preview overlay should become visible earlier or indicate to users that new partial gestures are available.

It was hard for users to see that a gesture composed of two partial gestures (for example, a circle followed a leftward line) can also be drawn at once. The group that started with gesture previews enabled was drawing composed gestures very slow, constantly waiting for new visual feedback to appear and not really understanding what they were doing. However, almost every user had a revelation at some point, saying things like 'Now I see it! I can add these preview parts together and they form a complete gesture!'. At that point they understood the point of our interface and were able to learn new gestures andd to draw gestures both with- and without gesture previews. The group that started with a handout with available gestures had this insight much quicker, since they had seen the complete gestures on paper already. In the end, all users understood the concept of composed gestures and the majority liked them, saying for example 'That's actually pretty useful!' and 'you can learn very complex gestures this way'.

7.4.5 Other observations

We observed that several users tried to cancel a partial gesture by drawing it backwards, like they were trying to erase it. This behavior was unexpected and it is not supported by our **GesturePreviews** framework. However, as multiple users tried this it might be a good addition to a gestural interface. Further research in this direction is desired.

None of the users found out by themselves that they could also browse through the gesture preview overlay by clicking on the labels displayed at the end of each path. However, when introduced to this possibility, several users liked it. This is reflected in the completed questionnaires:

I prefer to touch preview labels rather than drawing gestures.	
Agree Completely	1
Agree	2
Neutral	1
Disagree	3
Completely disagree	1

Three users liked the possibility of using gesture previews as a menu instead. The user that agreed completely with the statement was the user who did not like working with gestural interfaces and he took any opportunity to avoid drawing gestures. Why the other two users prefer to touch labels instead is not clear. Maybe they felt it was quicker or preferred this because they had troubles drawing gestures since they did not press the touch screen hard enough. In any case, it seems that the support for clicking labels is a nice feature but is not really important.

Finally, almost all participants mentioned, either in the questionnaire or during the video session, that they want more feedback from the application. Some of the suggestions we received are:

- Feedback in the form of sound whenever a gesture is completed, initiated or cancelled.
- A screen flash when the application switches from local- to global gesture mode and vice versa.
- Sound or an error message when a user has drawn a gesture that is not recognized by the application.
- Animations to clarify gesture paths and composed gestures.
- Tactile feedback, although that is not possible on most touch screens.

This seems to be another usability issue of our implementation and users argued that improving this could greatly increase the usability of gesture previews.

7.5 Analysis of recorded gestures

7.5.1 A word of warning

As discussed previously, the **GesturePreviews** framework recorded all user input. Our goal was to analyse this data afterwards to find out whether gesture previews affect the accuracy with which gestures are drawn. While analysing the recorded video footage we discovered an interesting side-effect of gesture previews: users appear to draw gestures more slowly when gesture previews are enabled. In an ideal scenario, we could analyse the recorded data to find out whether gesture previews significantly affect the speed and accuracy of drawn gestures.

However, as discussed previously our proof-of-concept implementation suffered from several usability issues. Because of that it was hard for participants to find out how to switch between local- and global gesture mode and it was challenging to understand the concept of composed gestures. Since we only had a limited amount of time to conduct our user study, we sometimes demonstrated to users how to interact with the application to speed up the learning process. An example can be seen in figure 7.2.

Figure 7.2: The author interfering with the user study.

In a few other occasions the authors interfered with the user study as well. This is unfortunate, because in doing so the user study was contaminated. The tabletop device on which the user study was conducted cannot detect which user is touching the screen. This means that the database with recorded gestures contains gestures drawn both by the participants and the authors of the user study, who already had significant experience with the available gestures (in fact, they have

designed them).

Another problem we discussed was the quality of the tabletop device we used. Its touch screen had to be pressed firmly in order for a touch to be registered. Participants of the user study did not know this. This led to the unfortunate situation that several gestures were only recorded partially, since the tabletop device stopped registering user input at some point because the user was not pressing the touch screen hard enough.

Besides this, due to the experimenting nature of the user study, participants experimented with drawing gestures in all sorts of ways. Quickly, slowly or even with their other hand. We cannot stress enough that the database with recorded gestures is not really suited for extensive analysis. We have yielded some results from the database but these should be acclaimed critically. A stricter and larger user study is necessary to reach more definite conclusions. Nevertheless, we present our analysis in the next section.

7.5.2 The effect of gesture previews

To find out how gesture previews affect the way participants draw gestures, we split the database with recorded gestures into two parts. The first half contains all gestures drawn with gesture previews enabled. The other half contains all gestures drawn with gesture previews disabled. We can already see some interesting distinctions when we look at the recorded gestures stored in these different databases.

For three of the gestures that were used in our user study, we selected seven random matches from both databases. The paths associated with these matches are given in figures 7.3, 7.4 and 7.5 [2]. On the left side you see seven random instances of a gesture drawn with gesture previews enabled. And on the right side you see seven random instances of the same gesture drawn with gesture previews disabled.

In figure 7.5 can be observed that the gestures drawn with gesture previews enabled look more consistent than their counterparts. And in figure 7.3 one can observe that the scale with which the initial circle is drawn varies much less when gesture previews are enabled.

Although these figures display only a small sample of all recorded gestures, they do suggest that gestures drawn with gesture previews enabled are more accurate. This hypothesis holds for the entire sample size. To show this we plotted the

[2] All gestures are translated to start at the origin of the image.

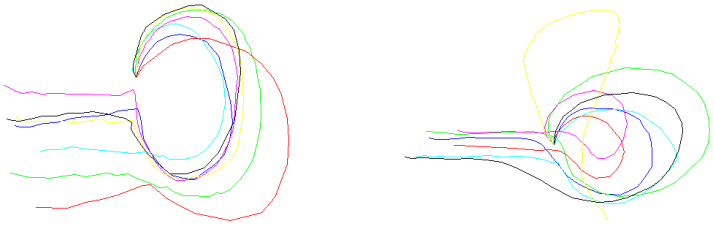

Figure 7.3: Gesture B.1 drawn with previews enabled (left) and disabled (right).

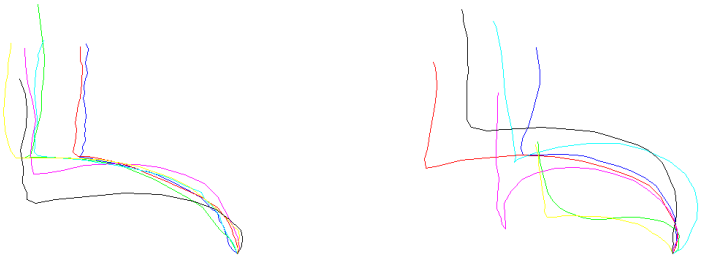

Figure 7.4: Gesture B.4 drawn with previews enabled (left) and disabled (right).

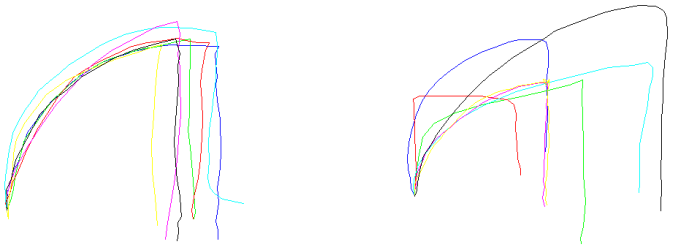

Figure 7.5: Gesture B.5 drawn with previews enabled (left) and disabled (right).

accuracy of drawn gestures against the duration it took to draw them. Figure 7.6 shows this plot for all gestures drawn without gesture previews and figure 7.7 shows this plot for all gestures drawn with gesture previews enabled. A lower 'accuracy' score means that a gesture is better recognized by the gesture recognizer. We see that the first set of gestures has an average score of about 8, while the second group has an average score of roughly 6.5. This means that, on average, gestures drawn with gesture previews enabled were better recognized by our framework.

During the analysis of the video footage we discovered that users tend to draw gestures more slowly if gesture previews are enabled. This effect is also visible in these plots. Drawing a gesture took on average around 200ms if gesture previews were disabled, and took roughly 300ms if gesture previews were enabled. Though this seems like a significant slowdown, it is not as much as we expected based on the analysis of the video footage.

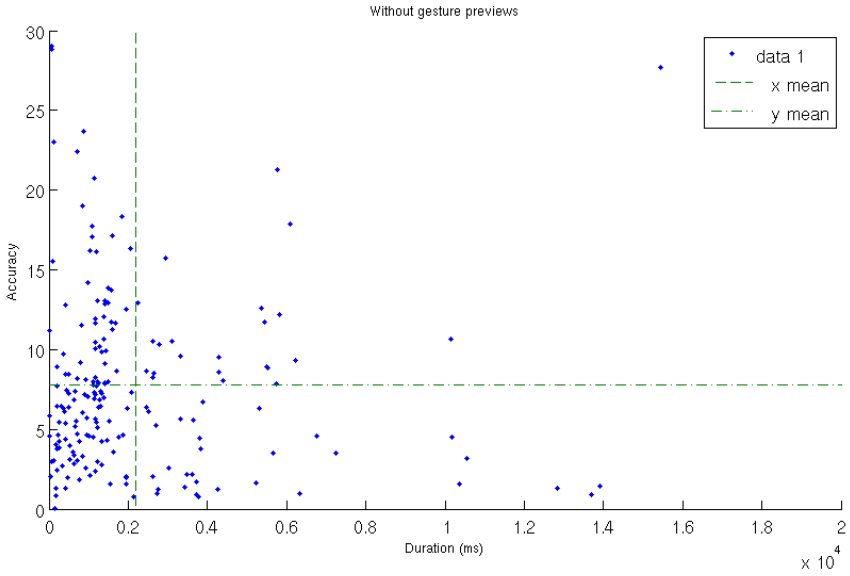

Figure 7.6: Plot of duration versus accuracy of gestures, when gesture previews are disabled.

A high percentage of gestures is drawn within a fraction of a second, according to these plots. This seems very fast. In the video footage we observed that users require more time to draw a gesture. We believe that one of the reasons for this discrepancy is that our database contains a lot of gestures that are only partially recorded due to participants not pressing the touch screen hard enough. We do not know the extent to which our database is filled with incomplete gestures. Again, we would like to urge the reader not to overestimate the significance of the analysis in this section.

We expected to see a correlation between the time it takes to draw a gesture and its accuracy. In other words, the slower a user draws a gesture, the more accurate we thought it would turn out. But in the plots given in figures 7.6 and 7.7 we see no significant relation between these two factors. We do observe that most gestures with an accuracy score $>= 10$ are drawn within 200ms. But again, these could be gestures that were partially recorded and thus we cannot use this observation to falsify or verify our hypothesis. Further research is required

to determine whether drawing speed affects gesture accuracy in a significant way.

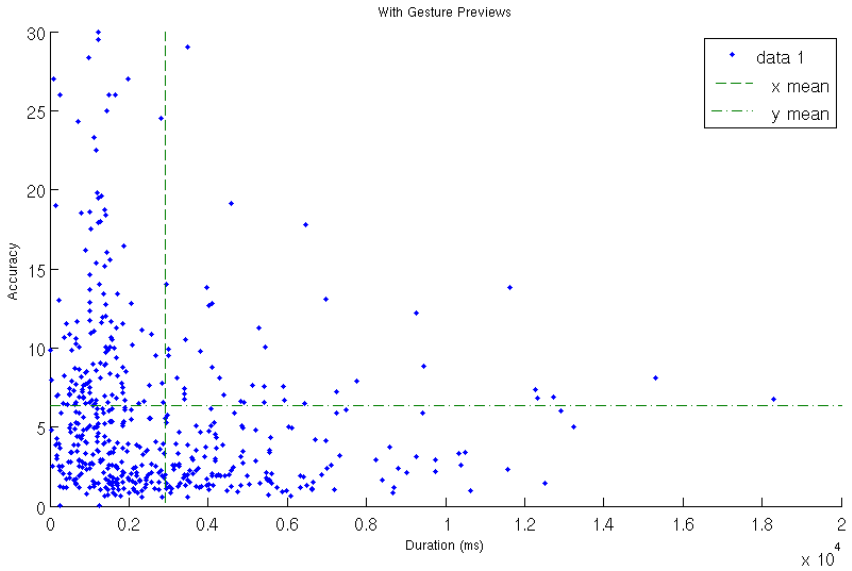

Figure 7.7: Plot of duration versus accuracy of gestures, when gesture previews are enabled.

Finally we would like to present figure 7.8 to the reader. This figure shows the points of origin of all gestures drawn during the user study. Clearly visible is that all users stood at the south side of the tabletop touch screen. Most gestures are drawn somewhat to the right of the center of the tabletop. We assume this is because most participants were right-handed, but are not sure since we have no information about the handedness of the participants. This plot nicely illustrates the concept of territoriality on tabletop touch screens. Users were free to touch the tabletop anywhere they liked, but strongly preferred the area right in front of them.

7.6 Conclusion

In this chapter we presented our user study and its results. The majority of the participants enjoyed working with our gestural interface and all participants think gestural interfaces are useful in certain contexts. The participants think that drawing gestures forms a fast an pleasant way to interact with computers. However, we have observed that both software and hardware should work fast and flawless. If they do not, gestural interfaces can quickly become an unpleasant experience.

The concept of gesture previews was relatively easily understood (and liked) by

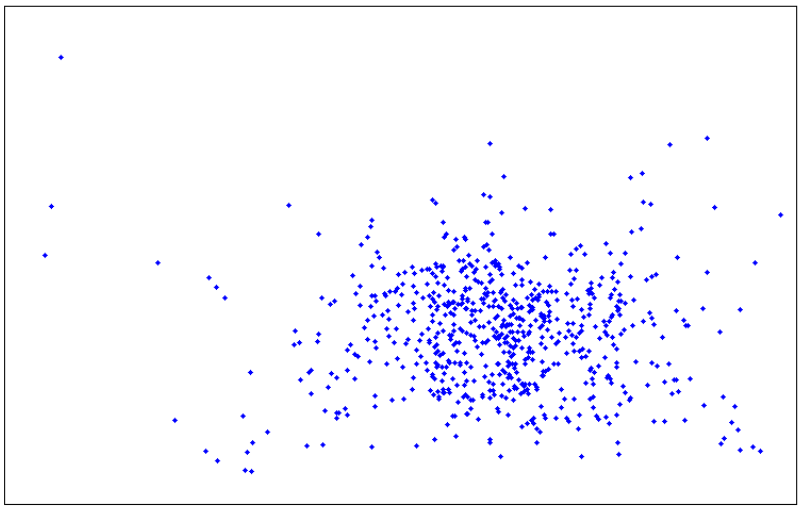

Figure 7.8: Points of origin of all gestures drawn during the user study.

most users. On the other hand, the concept of composed gestures was more difficult to grasp. Everything considered, participants were reasonable enthousiastic about gesture previews, despite the shortcomings of our implementation and the available tabletop touch screen.

Chapter 8

Conclusion

8.1 Introduction

In the this final chapter we analyze the results from our user study to see if gesture previews meet the three criteria introduced in the previous chapter. Ideally, gesture previews help novice users with drawing gestures, do not distract and/or limit expert users and increase the accuracy with which gestures are drawn.

We also look at ways in which our **GesturePreviews** implementation can be improved and discuss potential future research. After that we give our verdict about gesture previews.

8.2 Interpretation of user study

8.2.1 Gesture Previews for novice users

In our user study we observed that the participants, who had no relevant experience with gestural interfaces, liked worked with gestures and thought that gesture previews were a useful way to discover new gestures. However, as discussed in the previous chapter sometimes the authors interfered with the user study to show participants how the interface worked. Undoubtedly this makes learning to use the interface a lot easier and the question arises whether the participants would have understood gesture previews just as quick on their own.

This is reflected in the results of the questionnaire. When confronted with the statement 'Gesture previews increase the usability of gestural interfaces for novice users' participants reacted modestly positive. This is strange, since the participants (all novice users) personally were much more excited about gesture

previews! This is probably explained by the fact that we helped them with understanding the interface. In order for novice users to really benefit from gesture previews, we have to fix four main problems in our implementation.

First of all, gesture previews should fade in earlier and quicker at all times. This was suggested by almost all participants of the user study. Currently gesture previews fade in too late and too slow. Because of this, users fail to intuitively grasp the relation between the preview of a gesture and actually drawing it. Implementing this is not trivial, as the framework has to detect in real-time what gesture a user is drawing, but it should be possible. We believe that this will greatly increase the usability of gesture previews for novice users.

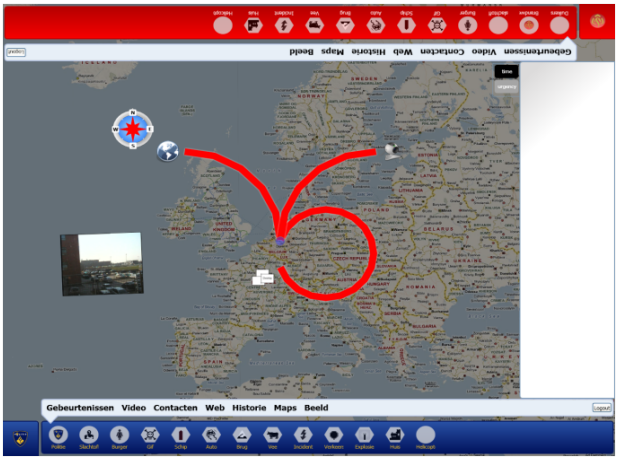

Figure 8.1: The interface is currently static, nothing is in motion.

The overlay with gesture previews should also use animation to convey the feeling of motion to the user. Currently the interface is static, always appearing as seen in figure 8.1. The available gesture paths should be animated in such a way that they make clear to users that they can follow the available paths with their fingers. The paths that are solid red lines in the current implementation could, for example, be replaced by a stream of arrows. Animations can also show users that a partial gesture can be followed by several other partial gestures, preventing users from lifting their fingers too quickly from the screen. This was something we observed a lot during the user study.

The distinction between local and global gestures was not very clear to the participants of the user study. The concept of touching a compass in order to switch to global gesture mode is not intuitive for novice users and none of the participants figured that out on their own. The other way to inititate a global gesture (touching the screen for a second in one place) is not very intuitive either. Perhaps an

application is better off using either only local gesture or only global gestures. This way users avoid the confusing switching of modes.

Finally, as discussed in the previous chapter, all users want more feedback from the application. Text and movies should indicate the meaning of a gesture more clearly. Participants want to hear sounds when they complete or cancel a gesture. Animations or screen flashes should appear after a gesture is drawn to show users that their gesture was recognized. Users also want feedback when they draw a gesture incorrectly or when the screen is 'locked' to draw a global gesture. All this feedback can, if used in a meaningful way, help users to learn working with gestural interfaces.

To conclude, in its current implementation, gesture previews are not very useful for novice users. Understanding what gesture previews are and how to benefit from them is difficult for novice users, unless helped by an expert. But our application is not very polished and we think that if it is improved in the four aforementioned ways gesture previews could become a useful help for novice users. A new user study is required to verify this.

8.2.2 Gesture Previews for expert users

If a complete gesture is drawn with reasonable speed, gesture previews never appear. This means that an expert user will never see gesture previews unless he wants to. Therefore, we conclude that gesture previews are not distracting or annoying for expert users. This was confirmed during the user study. The participants who learned gestures from a paper handout drawed them quickly and did initially not even notice the preview overlay.

Participants of our user study mentioned that gesture previews can be useful for advanced users that need to learn a new set of gestures in another context. This is interesting. We came up with the concept of gesture previews to help novice users, but apparently expert users could benefit from it as well.

As an example participants mentioned complex image manipulation. An example of such an application can be seen in figure 8.2. If ported to a touch screen, users could use gestures to select tools and manipulate images, remembering common gestures and using gesture previews to discover other available gestures.

8.2.3 Increase of accuracy

We have observed in our user study that when gesture previews are enabled, users tend to exactly follow the paths displayed on the screen. This results in gestures

Figure 8.2: Complex image manipulation. *(Image from Adobe)*

that are drawn precisely and always have the same size and aspect ratio. We can see this clearly in the video footage. Probably these gestures can be better classified by a gesture recognizer, but unfortunately our database of recorded gestures is not reliable enough to confirm this.

However, the fact that users draw gestures precisely when they see a gesture preview is not really interesting to us, since gesture previews are meant to be used as a learning tool. What we would really like to know is if users *continue* to draw gestures more accurate even after they stopped using gesture previews. In other words, do gesture previews 'teach' users to draw gestures more accurate? To measure this, a user study of several weeks is required. One group of participants learns gestures using gestures previews, and a control group does not. Afterwards we can compare the accuracy of gestures drawn over time by both groups.

We conclude that gesture previews help users with drawing gesture more accurate, but we do not know whether this effect is permanent.

8.3 Future work

In our user study we have seen some unexpected behavior. For example, users tried to undo a partial gesture by drawing it backwards. We have found no related work regarding this interesting behavior and as far as we know it is not supported in any touch screen interface. However, multiple users tried this during our user study, raising the question whether such a feature should be supported in the next generation of gestural interfaces.

Another observation made during the user study is that users tend to draw gestures much slower when gesture previews are enabled. This could be caused by

the fact that previews fade in too slow in our implementation, as was mentioned by several participants. On the other hand, this could be an unwanted side-effect that is inherent to gesture previews, i.e. there could be a psychological reason for it. When we asked a participant why he was drawing gestures slowly when previews were enabled he responded: 'I feel being guided by the previews'. As soon as gesture previews disappeared, participants started drawing gestures faster again. So it seems this effect is not permanent. We do not know why a displayed preview decreases user interaction speed. Neither do we know whether this problem is unique for gesture preview or also holds for other forms of visual feedback. An understanding this phenomenon might help to design more effective user interfaces.

Also interesting is what participants of the user study did *not* mention. While designing our interface we were worried about issues regarding rotation and orientation. During our user study participants only stood at one side of the table. In this case gesture previews were always oriented correcly since we only used the 'classic view' that was introduced in chapter 6. However, if a user were to stand on the opposite side of the tabletop, all content in the gesture previews overlay would appear upside-down.

When confronted with this problem, the participants expected it to be solved automatically by the software, thinking it was a trivial problem. Creating a tabletop interface that lives up this expectation and always orients gesture previews (and other interface components) the right way is challenging. An interesting new technique is shadowtracking. Echtler et al. [3] discuss an interface in which a camera registers shadows of users' hands and arms. From this data an algorithm can deduce where users are located around the table (see figure 8.3). This information can subsequently be used to position and orient interface components. Another solution is that users wear RFID-chips to provide a tabletop device with locational awareness. The usage of such techniques to orient interface components forms a starting point for future research.

Participants of our user study all mentioned that they want more feedback. We agree that a next version of our **GesturePreviews** framework should meet this request. However, the design of an interface is no hard science. Different users can even have a different opinion about a single interface. Interesting questions are:

- What kinds of audible feedback should be used, and when?
- What colors, sizes and transparency values do users like best for gesture previews?

Figure 8.3: Shadow tracking. The shadows on the touch screen are actually rendered by the computer.

- Which animations should be used to indicate that follow-up gestures are available?
- How to convey an error message when a gesture is not recognized?
- Is tactile feedback an option for future touch screens? And if so, in what form?

Finding answers to these questions is a process of trial-and-error. Using our framework the appearance of gesture previews can relatively easy be adapted. Experimenting is required to come up with an intuitive implementation that best serves its final goal: helping novice users with gestural interaction.

A prime point of interest that we, unfortunately, have not looked at is the use of gestures previews in a multi-touch environment. An interesting question is whether gesture previews can help and encourage users to perform multi-touch or even cooperative gestures. The **GesturePreviews** framework does not support this in its current state but it should be possible to add support for this. The **Touchlib** library we use for finger tracking already supports multi-touch input and our gesture recognizer can already match multi-touch gestures. The main problem is that the framework has to decide for every touch if a user wants to draw an independent gesture or if the user wants to join in on an existing multi-touch gesture. A possible solution to this problem is that the gesture preview overlay shows an area where users can join in on a multi-touch gesture. User

input that originates in this area is treated as part of the multi-touch gesture. All other user input is treated independently.

We think that the development of a game could serve as an excellent starting point for further research in this direction. Besides fun to play such a game could also serve a useful purpose. Piper et al. have developed SIDES [15], a cooperative tabletop computer game for social skills development. This game was used in therapy classes to encourage cooperation between adolescents with Asperger's Syndrome. Perhaps cooperative gestures can be used in a similar way.

Also interesting is the question whether complex multi-touch gestures are actually desirable in touch screen interfaces. As far as we know, they are not in widespread use, despite the emergence of multi-touch screens over the past few years. Perhaps users prefer other ways of interaction? A user study could lead to an answer to this question.

Finally, in the chapter on implementation we have discussed a file format for storing gestures and gesture previews. We have designed an XML format that met our requirements and is relatively easy to work with. But this format is designed without much analysis and is probably not an optimal solution. A follow-up study should look at the requirements of a gesture file format in more detail, and discuss the advantages and disadvantages of alternative solutions. In theory all stakeholders can benefit from a widespread standard for sharing and storing gestures. Libraries that use this standard can save work for developers who want to integrate gestures in a wide range of devices. And for end users such a standard might, in the long term, lead to a set of 'universal' gestures that they can use on all devices.

Currently, gestures are arbitrarily selected by software developers. Different devices and applications feature different sets of gestures. In contrast, on personal computers several important keyboard shortcuts are commonly accepted. For example, *control+c* and *control+v* are used in a wide range of applications to copy and paste data. We believe that a similar set of widely acknowledged gestures will increase the usability of gestural interfaces. A gesture file format might encourage the shift towards that situation.

On the other hand, such a standard might be cumbersome to work with and could be unable to capture the fluid nature of gestural interfaces. The development of other applications that use a generic gesture description format (for example a gesture preview design application) can serve as a starting point for further research is this direction.

8.4 Conclusion

In this thesis we have presented our work on gesture previews and the related concept of composed gestures. The prime goal of these concepts is to make gestural interfaces more intuitive for new users. During our user study we observed that the participants initially did not understand the interface of the emergency control room application. In this way gesture previews did not live up to our initial expectations.

However, we feel that this disappointment is mostly caused by the implementation of our **GesturePreviews** framework, which in hindsight is lacking in some areas. We reach this conclusion because all participants were enthousiastic about gesture previews and the possibilities they offered during the user study. This was also reflected in the completed questionnaires. Some participants were really excited about the gestural interface and gave large lists of possible enhancements for gesture previews.

From our user study we learned that users appreciate lots of feedback. In a pure gestural interface no mouse pointer is visible, no menus are visible and there are no buttons to click. Such an interface has to give lots of feedback to users when they try to interact, otherwise they get completely lost. Gesture previews are one form of feedback but they are not enough on their own. They should be accompanied by relevant sounds, animations and texts whenever something could be unclear for a novice user. This is something we underestimated beforehand.

An improved implementation of our **GesturePreviews** framework and a bigger user study are required to give a final verdict about gesture previews. Until then we remain moderately positive. It is our hope that gesture previews can be used to develop a next generation of interfaces that make working on touch screens easier and more intuitive.

Bibliography

[1] C. M. Brown. *Human-computer interface design guidelines*. Ablex Publishing Corp., Norwood, NJ, USA, 1988. [cited at p. 14]

[2] Kris Luyten Davy Vanacken, Alexandre Demeure and Karin Coninx. Ghosts in the Interface: Meta-user Visualizations as Guides for Multi-touch Interaction. In *Proc. IEEE Tabletop*, pages 87–90, Los Alamitos, 2008. IEEE Computer Society. [cited at p. 16, 27]

[3] Florian Echtler, Manuel Huber, and Gudrun Klinker. Shadow tracking on multi-touch tables. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*, pages 388–391, New York, NY, USA, 2008. ACM. [cited at p. 67]

[4] Alan Esenther and Kathy 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. [cited at p. 15]

[5] Clifton Forlines, Daniel Wigdor, Chia Shen, and Ravin Balakrishnan. Direct-touch vs. mouse input for tabletop displays. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 647–656, New York, NY, USA, 2007. ACM. [cited at p. 15]

[6] Jens Grubert, Sheelagh Carpendale, and Tobias Isenberg. Interactive stroke-based NPR using hand postures on large displays. Technical Report 2007-883-35, Department of Computer Science, University of Calgary, Canada, December 2007. [cited at p. 21]

[7] Jefferson 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. [cited at p. 10]

[8] Mark S. Hancock, Sheelagh Carpendale, Frederic D. Vernier, Daniel Wigdor, and Chia 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. [cited at p. 17]

[9] Uta Hinrichs, Sheelagh Carpendale, and Stacey D. Scott. Evaluating the effects of fluid interface components on tabletop collaboration. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 27–34, New York, NY, USA, 2006. ACM. [cited at p. 17]

[10] Uta Hinrichs, Mark S. Hancock, M. Sheelagh T. Carpendale, and Christopher Collins. Examination of text-entry methods for tabletop displays. In *Tabletop*, pages 105–112, 2007. [cited at p. 14]

[11] Martin Kaltenbrunner, Till Bovermann, Ross Bencina, and Enrico Costanza. Tuio - a protocol for table based tangible user interfaces. In *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005)*, Vannes, France, 2005. [cited at p. 40]

[12] Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 258–264, New York, NY, USA, 1994. ACM. [cited at p. 28, 33]

[13] Hyeon-Kyu Lee and Jin H. Kim. An hmm-based threshold model approach for gesture recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):961–973, 1999. [cited at p. 38]

[14] Meredith Ringel Morris, Anqi Huang, Andreas Paepcke, and Terry Winograd. Cooperative Gestures: Multi-User Gestural Interactions for Co-located Groupware. In *Proc. CHI*, pages 1201–1210, New York, 2006. ACM Press. [cited at p. 22, 25]

[15] Anne Marie Piper, Eileen O'Brien, Meredith Ringel Morris, and Terry Winograd. Sides: a cooperative tabletop computer game for social skills development. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 1–10, New York, NY, USA, 2006. ACM. [cited at p. 69]

[16] James A. Pittman. Recognizing handwritten text. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 271–275, New York, NY, USA, 1991. ACM. [cited at p. 38]

[17] Adrian Reetz, Carl Gutwin, Tadeusz Stach, Miguel Nacenta, and Sriram Subramanian. Superflick: a natural and efficient technique for long-distance object placement on digital tables. In *GI '06: Proceedings of Graphics Interface 2006*, pages 163–170, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society. [cited at p. 17]

[18] Dean Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25(4):329–337, 1991. [cited at p. 38]

[19] Stacey D. Scott, Karen D. Grant, and Regan L. Mandryk. System Guidelines for Co-located, Collaborative Work on a Tabletop Display. In *Proc. ECSCW*, pages 159–178, Norwell, MA, 2003. Kluwer Academic Publishers. [cited at p. 25]

[20] Stacey D. Scott, M. Sheelagh, T. Carpendale, and Kori M. Inkpen. Territoriality in collaborative tabletop workspaces. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 294–303, New York, NY, USA, 2004. ACM. [cited at p. 17]

[21] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. UIST*, pages 159–168, New York, 2007. ACM Press. [cited at p. 38]

Appendices

Appendix A

A sample gesture definition in XML

```
<?xml version="1.0" encoding="utf-8"?>
<root>
<segmentDeclaration name="line" uniformScaling="false">
<point x="0" y="0" />
<point x="0" y="0,1" />
</segmentDeclaration>

<segmentDeclaration name = "halfcircle"
  uniformScaling = "false">
<point x="0" y="0" />
<point x="0,0618034" y="0,009788692" />
<point x="0,117557" y="0,03819659" />
<point x="0,1618034" y="0,08244295" />
<point x="0,1902113" y="0,1381966" />
<point x="0,2" y="0,2" />
<point x="0,1902113" y="0,2618034" />
<point x="0,1618034" y="0,317557" />
<point x="0,117557" y="0,3618034" />
<point x="0,0618034" y="0,3902113" />
<point x="0" y="0,4" />
<point x="-0,0618034" y="0,3902113" />
<point x="-0,117557" y="0,3618034" />
</segmentDeclaration>
```

```
<gesture name="republican">
<segment name="line" angle="0,5"/>
<segment name="line" angle="-0,5"/>
<segment name="halfcircle" angle="0,0"/>
<preview source="elephant.png"
  relativeWidth="0,05" relativeHeight="0,05" />
</gesture>


<gesture name="democrat">
<segment name="line" angle="4,71"/>
<segment name="line" angle="0"/>
<preview source="mirror.png"
   relativeWidth="0,05" relativeHeight="0,05" />
</gesture>

<gesture name="third">
<segment name="line" angle="0,3"/>
<segment name="line" angle="0,6"/>
<segment name="line" angle="0,9"/>
<segment name="line" angle="1,2"/>
<segment name="line" angle="1,5"/>
<preview source="path.png"
  relativeWidth="0,1" relativeHeight="0,1" />
</gesture>

</root>
```

Appendix B

Gestures in the emergency control room

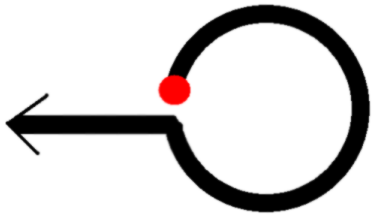

Figure B.1: First gesture: change map type to 'street view'.

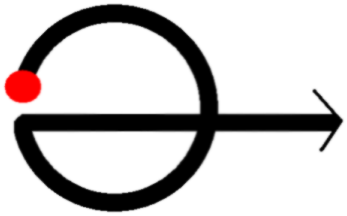

Figure B.2: Second gesture: change map type to 'satellite view'.

Figure B.3: Third gesture: zoom in on Boston, USA.

Figure B.4: Fourth gesture: zoom in on Groningen, the Netherlands.

Figure B.5: Fifth gesture: point the webcam to an office space.

Figure B.6: Sixth gesture: point the webcam to a parking lot.

Appendix C

Gesture Previews Questionnaire

C.1 Vragenlijst voorafgaand aan experiment.

De resultaten van deze vragenlijst alsmede opgenomen videobeelden kunnen worden verwerkt in mijn scriptie en gebruikt worden voor mogelijk andere wetenschappelijke doeleinden.

Plaats je handtekening als je het hier mee eens bent.

Bij meerkeuzevragen: omcirkel het antwoord dat op jou van toepassing is. 1 = helemaal mee eens, 2 = mee eens, 3 = neutraal / geen mening, 4 = niet mee eens, 5 = helemaal niet mee eens.

Stelling 1: Omcirkel de stelling die het beste bij jou past.
1. Ik heb geen ervaring met tabletop touch-screens.
2. Ik heb wel eens gewerkt met tabletop touch-screens.
3. Ik werk regelmatig met tabletop touch-screens.

Vraag 2: Weet je wat een gestural interface is? Zo ja, probeer een korte omschrijving te geven.

C.2 Vragenlijst na experiment.

Bij meerkeuzevragen: omcirkel het antwoord dat op jou van toepassing is. 1 = helemaal mee eens, 2 = mee eens, 3 = neutraal / geen mening, 4 = niet mee eens, 5 = helemaal niet mee eens.

Stelling 3: Ik weet nu wat een gestural interface is.

1 2 3 4 5

Vraag 4: Heb je eerder gewerkt met een system dat gestures gebruikt. Zo ja, welk(e) syste(e)em(en)?

Vraag 5: Wat zie je als voordelen van een gestural interface ten opzichte van een traditionele interface? En wat zie je als nadelen? Je hoeft hier niet uitgebreid op in te gaan, enkele steekwoorden zijn voldoende.

Stelling 6: Ik denk dat voor nieuwe gebruikers een gestural interface makkelijker is te gebruiken dan een traditionele touch-screen interface met knoppen en menus.
1 2 3 4 5

Stelling 7: Ik vind het werken met een gestural interface aangenaam.
1 2 3 4 5

Stelling 8: Ik denk dat er specifieke touch-screen toepassingen zijn waar een gestural interface toegevoegde waarde heeft.
1 2 3 4 5

Vraag 9: Zo ja, aan welke toepassingen denk je dan?

Stelling 10: Ik snap het principe van gesture previews.
1 2 3 4 5

Stelling 11: Ik denk dat het voor gebruikers zonder multi-touch ervaring snel duidelijk is wat gesture previews zijn en hoe ze gebruikt kunnen worden.
1 2 3 4 5

Stelling 12: Ik denk dat voor gebruikers met multi-touch ervaring snel duidelijk is wat gesture previews zijn en hoe ze gebruikt kunnen worden.
1 2 3 4 5

Stelling 13: Ik vind het uitdelen van een papier met daarop beschikbare gestures een betere manier om gestures te leren kennen dan het gebruikmaken van gesture previews.
1 2 3 4 5

Stelling 14: Ik vind het makkelijk dat gesture previews ook als menu gebruikt kunnen worden door op de labels te klikken.
1 2 3 4 5

Stelling 15: Ik klik liever op de labels dan dat ik een gesture helemaal teken.
1 2 3 4 5

Vraag 14: Zie je nog verdere mogelijkheden om het principe van gesture previews te verbeteren? Of heb je nog andere op- of aanmerkingen? Plaats deze hier.

List of Figures

1.1 Gestural interaction in a golfing game. *(Image from Nintendo)* 3
1.2 Gestural interface in a science fiction movie. *(Image from 20th Century-Fox Film)* 4
1.3 Touch screen used in the reporting of the American presidential election in 2008. *(Image from CNN)* 5

2.1 A ticket machine with a touch interface 7
2.2 This gaming device has a resistive touch screen. *(Image from Nintendo)* 8
2.3 A popular mobile device with a capacitive touch screen. *(Image from Apple)* 9
2.4 Total internal reflection is disturned when a finger touches the screen. 10
2.5 Frustrated reflection as captured by an infrared camera. 10
2.6 FTIR tabletop touch screen available at TNO, Groningen. 11

3.1 An interface that is not very useful on a touch screen. *(Image from Microsoft)* 14
3.2 On-screen keyboard. *(Image from Apple)* 15
3.3 Personal assistant in a text editor. *(Image from Microsoft)* 16
3.4 Screenshot of *TouchGhosts* demonstrating how to zoom in on a photo. 16
3.5 A fluid interface: components flow around the border of the screen . . 17

4.1 A gesture that can be interpreted in multiple ways 19
4.2 Posture based rendering 20
4.3 A common multi-touch gesture *(Image from Microsoft)* 21
4.4 An example of a cooperative gesture. 22
4.5 Drawing for fun. 24
4.6 A gesture without an intrinsic meaning 25

5.1 Marking menu. *Image from Stephane Huot* 29
5.2 Untangling paths can be challenging 30

5.3 Two gestures that start in a similar way. 31
5.4 An example tree of partial gestures. 32

6.1 Emergency control room software. *Image from of TNO* 35
6.2 Orientation of gesture previews in classic view. Suitable for most devices. 39
6.3 Orientation of gesture previews in center view. Useful when multiple users stand are working on a tabletop device. 40
6.4 Screenshot of Touchlib. *Image from NUI group* 41
6.5 Screenshot of an OpenGL implementation of the rendering subsystem. 42
6.6 Emergency control room application, showing real-time video footage *Image from of TNO* 43
6.7 Initial overlay with previews. *Image from TNO* 44
6.8 The overlay fades out when a user is drawing a gesture. *Image from TNO* 45
6.9 A new set of gesturepreviews. *Image from TNO, Groningen* 46
6.10 No overlay appears when a complete gesture is drawn at once. *Image from TNO* 47

7.1 Video footage of user study. Users stay anonymous. 51
7.2 The author interfering with the user study. 57
7.3 Gesture B.1 drawn with previews enabled (left) and disabled (right). . 59
7.4 Gesture B.4 drawn with previews enabled (left) and disabled (right). . 59
7.5 Gesture B.5 drawn with previews enabled (left) and disabled (right). . 59
7.6 Plot of duration versus accuracy of gestures, when gesture previews are disabled. 60
7.7 Plot of duration versus accuracy of gestures, when gesture previews are enabled. 61
7.8 Points of origin of all gestures drawn during the user study. 62

8.1 The interface is currently static, nothing is in motion. 64
8.2 Complex image manipulation. *(Image from Adobe)* 66
8.3 Shadow tracking. The shadows on the touch screen are actually rendered by the computer. 68

B.1 First gesture: change map type to 'street view'. 79
B.2 Second gesture: change map type to 'satellite view'. 79
B.3 Third gesture: zoom in on Boston, USA. 80
B.4 Fourth gesture: zoom in on Groningen, the Netherlands. 80
B.5 Fifth gesture: point the webcam to an office space. 80
B.6 Sixth gesture: point the webcam to a parking lot. 81