Gestures for pointing devices in screen-based environments

Florian Weil, www.derhess.de

December 4, 2010

Abstract

This paper analyses gesture design for pointing devices in screen-based environments. By exploring design patterns the analysis investigated the gesture design of five different end-user products: Desktop operating systems, mobile operating systems, 3rd Party software, small software products, and common hardware products. The beginning of the paper defines what a gesture is, and the various kinds of gestures. Afterwards the analysis merges the gesture design results with the basic commands for pointing devices. This approach points out which gestures are often used, and in which context they are used. The results give interaction designers and software engineers a guide for implementing gestures in their own products. Furthermore, the paper proposes solutions for gesture documentation, and a conceptual framework for complicated gestures. The last section takes an industrial design perspective on pointing devices as an input channel. It discusses the evolution of interface design from a hardware driven to a software driven approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interaction designers and software engineers have endeavoured to integrate gesture-based interaction in software products for years. With the advent of multi-touch screens and the launch of the iPhone gesture-based interaction has established new methods of interaction for Human-Computer-Interaction. Unfortunately, gestural-based interaction for Human-Computer-Interaction is quite different relative to traditional interaction via a mouse device. Interaction designer and software engineers are required to learn how gestures function, which concepts base gestures on, when it makes sense to use gestures, and in which cases gestures can harm the Human-Computer-Interaction.

The focus of this paper is on gestures for pointing devices in screen-based environments. This area of pointing devices most commonly relates to multi-touch screens, mouses, trackpads, tablets, and pen systems as input devices. This paper is divided in 5 chapters. In chapter 2 the word "gesture" and the context in which this paper uses the word "gesture" is defined. The last section outlines a functional command set for pointing devices aimed at interaction designers and software engineers to create new usable gestures.

This paper does not investigate free-form gestures via gloves or camera-based interaction. Furthermore, this paper only explores pointing devices, which are used in screen-based environments. That includes small (e.g. smart phones), medium (e.g. tablet devices), and Desktop Screens. Big city screens or scalable multi-touch screen systems like MultiTouch Cell[1] are not part of this investigation.

Chapter 3 contains gesture analysis results of current software and hardware products. The analysis investigated gestures used for Desktop Operation Systems, Mobile Operation Systems, 3rd Party software, small software tools, and additional input devices. The analysis reveals what type of interaction gesture patterns exist for these different products. The lack of gesture documentation

[1] MultiTouch Cell from Multitouch website, http://multitouch.fi/products/cell/, accessed March 2010.

and guidance systems for more complicated gestures is the focus of section 3.4.

Another important point of the paper is the evolution of input devices from an industrial design perspective. Chapter 4 puts forward the hypothesis that *"Our input devices will become more abstract in the future, and software (the virtual part) will define the rules of interaction."* Software is going to play a more important role than the hardware design. Smartphones with a Touch-Screen and have less than four buttons. These generation of smartphones are an evident for this evolution. Chapter 4 concludes by discussing the emergent issues between organic interfaces [2] and gesture-based interaction. The thesis will be falsified or confirmed, depending on the research results from some research and development labs.

The conclusion of results aims to contribute and further research in the field of gestural interaction. For instance, how various media and screen environments influence gesture-based interaction.

[2]Organic User Interface are user interfaces with non-planar displays that may actively or passively change shape via analog physical inputs, see also http://www.organicui.org

Chapter 2

Gestures for pointing devices

The word "gesture" is a broadly defined term. It is often used colloquially and various scientific disciplines use the word "gesture" in different contexts. Therefore it is important to achieve a suitable and well defined concept for the word "gesture", which is usable for Human-Computer-Interaction. This is especially important in the context of pointing devices in screen-based environments.

2.1 Definition of gesture in Human-Computer-Interaction

The definition of gesture articulated by Kurtenbach and Hulteen is useful for defining a gesture-based interaction in Human-Computer Communication systems:

> "A gesture is a motion of the body that the contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on it's way to hitting a key is neither observed nor significant. All that matters is which key was pressed."[10]

Kurtenbach and Hulteen state that the movement for pressing a key on a keyboard is not a gesture, because the movement is not captured. Pointing devices have the ability to capture physical movements. Also the definition of a gesture from Dan Saffer supports that a gesture is a movement.

> "A gesture, for the purposes of this book, is any physical movement that a digital system can sense and respond to without the aid of a traditional pointing device such as a mouse or stylus. A wave, a head nod, a touch, a toe tap, and even a raised eyebrow can be a gesture."[17, page 2]

There is no definition of a gesture available which excludes pointing devices that are extensions of the body. Furthermore, pointing devices meet the requirements of a gesture recognition system defined by Dan Saffer:

> "Every device or environment that employs gestures to control it has at least three general parts: a **sensor**, a **comparator**, and an **actuator**. These three parts can be a single physical component, or, more typically, multiple components of any gestural system, such as a motion detector (a sensor), a computer (the comparator), and a motor (the actuator)."[17, page 13+14]

In the end we know every gesture is a combination of movement and a list of captured data (events). For this reason gestures in this paper consist of at least **2 events data**. For instance, a *mouse-button-press* is an event. A *mouse-button-press* event in combination with a quick following *mouse-button-release* event is already a gesture, which is called *mouse-click* gesture. A further 3-event-data gesture could be "*mouse-button-press + mouse-move + mouse-button-release*". This gesture belongs to the traditional Drag & Drop gesture. How a system reacts to different events (gestures), depends on interaction design. The interaction and interface designer is responsible for effective and intuitive gesture interaction.

2.2 Types of gestures

Bill Buxton[2] summarized the most important categories of gestures for Human-Computer Interactions. The first category is founded on Cadoz (1994). He grouped gestures into three types [2, page 2] :

semiotic communicate meaningful information

ergotic manipulate the physical world and create artefacts

epistemic learn from the environment through tactile or haptic exploration

Humans like to interact directly with objects [17, page 17]. For this reason ergotic and epistemic gestures can be a very good inspiration source for gesture design with pointing devices. We can adapt or transform these haptic gestures into the screen-based environment. For instance, pushing up and down the volume slider, works perfectly with a physical input and in a screen-based environment via drag interaction, as well.

In addition to these gestures, Rime and Schiaratura (1991) also specified a useful gesture taxonomy[2, page 2+3]:

symbolic gestures These gestures have a same meaning in each culture. For instance, the thumbs up hand sign for everything is "*OK*".

deictic gestures The gestures of pointing, or focusing the attention to specific events or objects in the environment. These gestures have a strong command character and therefore they are already well established in Human-Computer-Interaction.

iconic gestures These gestures communicate information about the size, shape or orientation of the objects. These gestures have a descriptive and an additional explanation character.

pantomimic gestures These gestures show the use of movement or how a object is used by an human. The human normally do a pantomimic gesture for emphasizing the done activity.

For our approach of utilizing pointing devices for gestural input the deictic and iconic gestures are the best gestures forms. Deictic gestures are well established and extensively explored in traditional command-based communication between human and computer systems. Therefore interaction designers must focus more on iconic gestures. Iconic gestures relate to the Direct-Object-Manipulation concept, which is nowadays used software. Direct-Object-Manipulation enables the user to change the size, form or other properties of a digital object directly. Current software products most commonly use the "Drag & Drop" metaphor for Direct-Object-Manipulation. With the advent of multi-touch screens the interface designs have an entire new array of possibilities for the Direct-Manipulation concept in screen-based environments. For instance, transform operations (scaling, rotation, etc.) at the same time. Iconic gestures explore such new possibilities in the best way, due to their related descriptive and explanatory character.

Moreover, Dan Saffer defines gestures as static gestures and dynamic gestures[17, page 179]:

Static gestures belong to the taxonomy of symbolic gestures. Meaning static gestures are performed and held like postures.

Dynamic gestures are a movement over time, like a "Drag & Drop" gesture or using the scroll wheel of the mouse.

In addition to the mentioned categories, this paper introduces a new category. This new category combines all these other categories, and distinguishes between intuitive, pre-trained and trained gestures. Transitioning fluidly between intuitive, pre-trained and trained gestures.

Intuitive gestures are obvious for the user. The user does not need to learn them. For instance, the selection process of an interface item. The user has to move the cursor to an object and then click on the object. This selection action functions the same way in the real world. Go to the object and grab/touch it. The user does it intuitively.

Pretrained gestures are gestures from predecessor interactive systems. For instance, the double click is not an intuitive and obvious gesture, but the

most people have already learnt this gesture from interaction with the mouse device.

Trained gestures are the most complicated gestures. Trained gestures are absolutely not obvious for the user. The user must learn them through guidance from a documentation or an instructor. In the most cases only expert users of a software use trained gestures, which could be compared with keyboard shortcut commands. Trained gestures can cause problems and confusion for the end-user, which will be discussed in detail in section 3.4.

In summary we have explored a variety of definitions for gestures and how they can be categorized. These different categories simplify the search for new usable gestures: defining the scope of gestural interaction and how we might develop new gestures. Not every gesture is useful, thus this section ends with an enlightening mnemotechnic verse from Dan Saffer [17, page 38]:

> "The more complicated the gesture, the fewer the people who will be able to perform it."

2.3 Commands for gestures

So far we have discussed mainly what a gesture is and the various categories of gestures that exist. This knowledge is important but almost useless without understanding it in the context of screen-based user interfaces. Every gesture is connected with a specific function or command. It is therefore important to understand the hierarchies and principle structures of current computer command communication. Gesture based interaction will not change this concept of communication with a screen. Gesture based interaction is able to perform tasks faster and more intuitivley for the user. Alan Cooper specified a very good structure model of computer command communication. This model is called "Canonical Vocabulary" and it is valid for all screen-based user interfaces[4, page 281]. He divided the commands in three different layers.

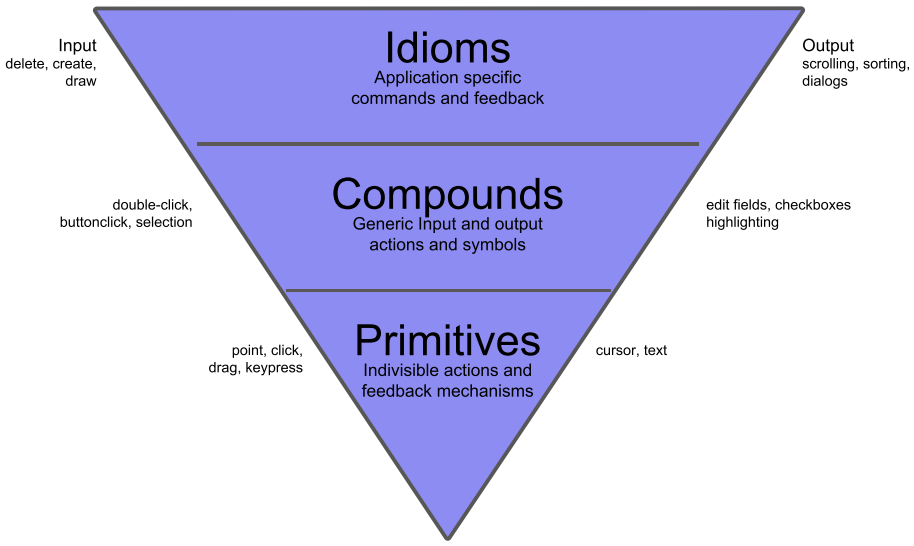


Figure 2.1: Canonical Vocabulary by Alan Cooper

This pyramid contains some interesting commands and gestures. From the command perspective almost every software supports delete, create, sorting and scrolling functions. So we can notice that these kinds of commands are the absolute basic commands in current screen-based software products.

From the gesture perspective the basics are *button-click*, *double-click*, *selection*, *drag* and *point*. All these gestures are already implemented in the operating systems. Every developer and Interaction Designer can work with this gestures without any concerns. These gestures are mostly obvious (e.g. Drag & Drop) or they belong to the pretrained gesture category (e.g. *double-click*).

Additional to the structure model of basic commands and gestures for screen-based user interfaces, Alan Cooper defined seven categories for Direct-Manipulation with pointing devices. These categories offer a useful overview of the command set for manipulating digital objects[4, page 377].

1. Pointing
2. Selection
3. Drag & Drop
4. Control manipulation
5. Palette tools
6. Object Manipulation (such as positioning, shaping, and resizing)
7. Object connection

The first four categories are already well explored and sophisticated. Only some minimal improvements are still possible. For instance, the command grouping object has already a function, but it could be better and more intuitive implement with the help of gestures. For the category "Palette tools" and "Object Manipulation" exist only an inconvenient set of functions. Especially, multi-touch systems deliver a more intuitive interaction form for changing the position, the shape and the size of digital objects. The last category "Object connection" exist already in project management or mind map software tools. The new possibilities of using more than one pointing input enables new interaction forms between different digital objects at the same time. Therefore this category "Object connection" is very interesting for gesture based interaction.

Having defined gestures and command communication in screen-based environments for pointing devices in a theoretical manner, the next chapter is going to explore how gestures are implemented and used for end-user products.

Chapter 3

Analysis of gesture based Interaction for end-user products

The following analysis seeks to find patterns in gestural interaction in current end-user products. After generating a list of used gestures, this chapter will explore regularly used gestural patterns. These patterns will offer an insight into which gestures are well established and which are not. From a user experience perspective it is important to reveal and utilise such patterns. Users will try to use gestures from one product in another product. If this transformation fails the user experience is broken. From a users perspective this assumption usually results in frustration with the interaction design. A broken user experience often triggers an aversion in using gesture based interaction.

Furthermore, analysis gives interaction designers and software engineers insights into how to further develop gesture based interaction in screen-based environments (especially in multi-touch screen environments).

The section 3.1 explains the analysis model in a more detail. How it applies the Bill Buxton idea of "Degree of Freedom" [3] for input devices. The subcategories are generated on this base of the "Degree of Freedom" theory. The sections 3.2 and 3.3 discuss the results on each level in detail. Each subcategory from a design and technical view has their own special characteristic. Analysis will explain these special characteristics and compare them to each other.

3.1 Concept and analysis model

The analysis model is comprised of two parts. One part summarizes the categories of gestures and the other summarizes the categories of software and hardware products.

The **categories of gestures** are related with the theory "Degree of Freedom"

from Bill Buxton[3]. He explains his theory:

> "The richness of interaction is highly related to the richness/numbers of degrees of freedom (DOF), and in particular, continuous degrees of freedom, supported by the technology. The conventional GUI is largely based on moving around a single 2D cursor, using a mouse, for example. This results in 2DOF. If I am sensing the location of two fingers, I have 4DOF, and so on."

Our analysis simplifies Bill Buxton's concept by basing it on the number of pointers. In Buxton's definition the mouse has two Degrees of Freedom, because it moves inside a two-dimensional world. In our concept the mouse cursor is a level 1 device, because it has only one point. A two finger interaction on a screen denotes 4 degrees of freedom in Bill Buxton's definition. In our case a two finger interaction is defined as a level 2 gesture, as it has two pointers. We simplified our concept because all the investigated input devices use only two dimensions (x and y coordinates) for interaction. In the case of Free-Form gestures and interaction in a three dimension environment this approach becomes invalid.

Every gesture level has subcategories with a different amount of movement. The analysis model distinguishes between one-way movement, two-way movement and n-way movement. In the figure3.1 you can see examples of each subcategory. The circle of these icons shows the starting point of the movement.

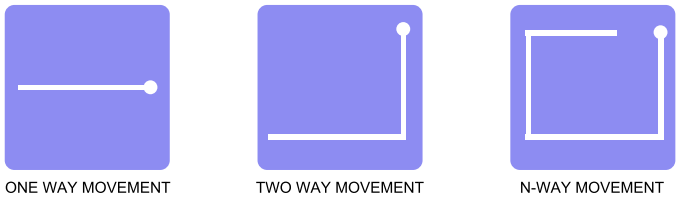


Figure 3.1: Movement categories of each gesture level

The **software and hardware part** includes products for the end-user and mass market. The software part has four different categories Desktop Operating Systems, Mobile Operating Systems, 3rd Party software, and additional software. The hardware category is defined by itself. A list of all investigated products can be viewed in figure 3.1.

The 3rd Party Tools present the most available Web Browser Plug-Ins. Many Internet Application (RIAs) are using these Plug-Ins. Therefore it will become important to consider, how these tools deal with gesture based interaction. Middle-ware Tools and libraries like reacTIVision[1],NUI Group Library[2],

[1]reacTIVision. From sourceforge website, http://reactivision.sourceforge.net/, accessed February 2010

[2]Touchlib. From NUI Group website, http://nuigroup.com/touchlib/, accessed February 2010

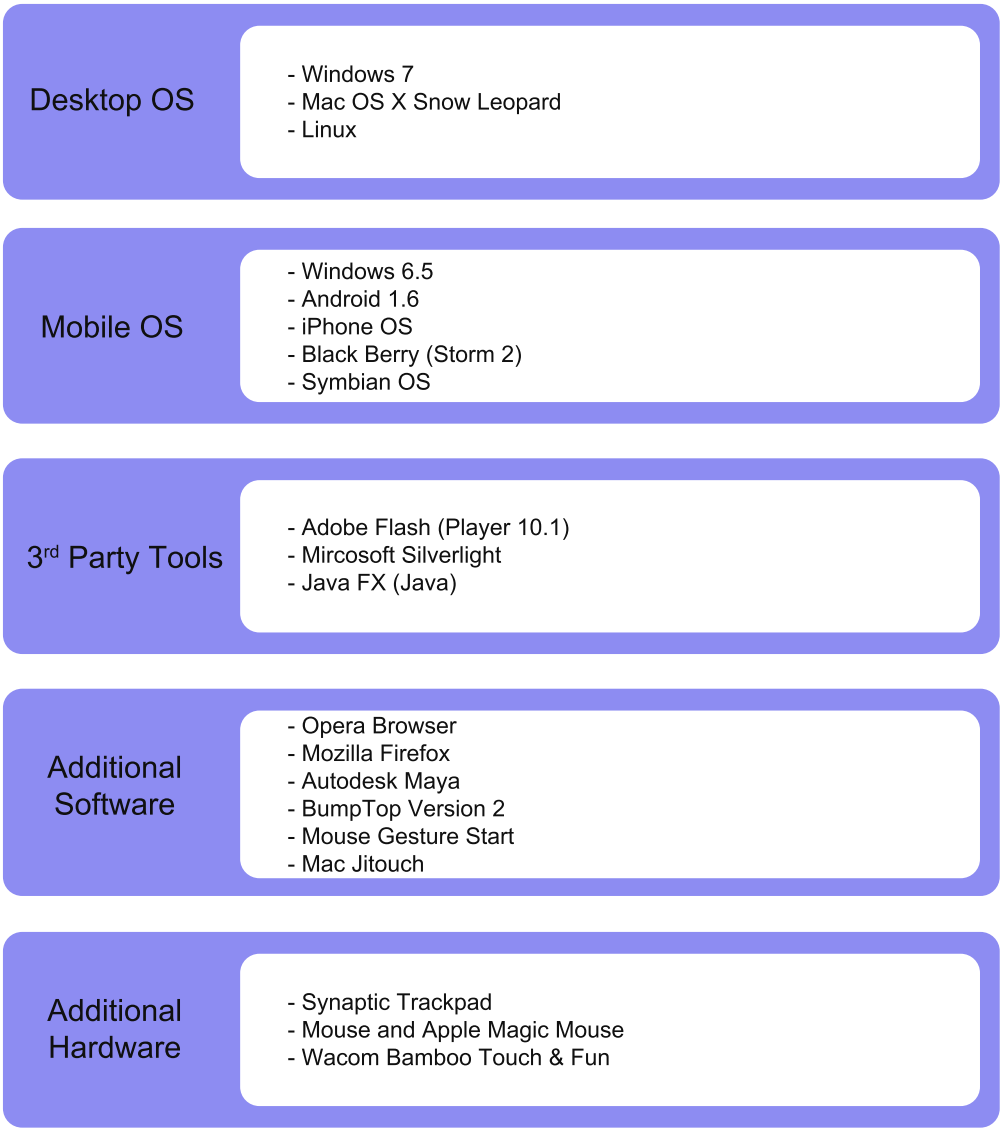


Figure 3.2: Overview of all investigated products

SPARS-UI[3], and MT4j[4] are not included in this analysis. These tools are normally used in installation systems or in special research environments. The libraries are important for prototyping and realizing new multi touch systems, but they do not play a big role for the mass market. The last software based category "additional Software" contains smaller software products which use gesture based interaction. The Internet browsers Mozilla Firefox and Opera as well as Autodesk Maya, BumpTop Version 2, MouseGesture Start, and Mac Jitouch belong to this category. BumpTop, MouseGesture Start and Mac Jitouch are characterised as extensions for the operating system rather than being an unique software product. Because of the their extension characteristic and non-compliant gesture design they afford the potential for innovation. A further motivation was to investigate the differences in gesture based interaction for big and small software products.

The hardware-based category explores the domain of common input devices for screen-based computer environments. Some of these devices extend the common hardware functionality of existing computer or notebook systems. Such devices are readily available in computer shops.

Having declared our the analysis model, the next section discusses the results of this qualitative research.

3.2 Results from a Interaction Design view

3.2.1 Level 1 gestures

The analysis revealed that level 1 gestures (only one pointer) are the most used gestures in current end-user products. The main reason for this fact is that the mouse is the traditional pointing device for screen-based software products, and it belongs completely in the level 1 category. *Click / Tap*, *double Click / double Tap*, and one-way movements are the most supported gestures in this category. These three interaction forms are almost used twice as often as another interactions forms.

During the analysis process it was not possible to define a common pattern for using two-way Movement gestures. Seven software products support this kind of gesture, but every product implements the gesture interaction in their own way. There exist no obvious interaction design convention for two-way gestures. The most systems (e.g. Google Android, Windows Mobile 6.5 etc.), which supports two-way and n-way movement interaction, deliver an additional Gesture Manager Utility Manager Software for their Programming SDKs. These tools simplify the creation process of custom gestures. Interaction designers and software engineers can benefit from such utility tools. They can define the form of the gesture, and then the gesture recognition software implements the detection algorithm and the gesture threshold automatically. Furthermore, no other

[3]Sparsh UI. From Google Code website, http://code.google.com/p/sparsh-ui/, accessed February 2010

[4]MT4j. From MT4j Wiki website, http://www.mt4j.org/, accessed February 2010

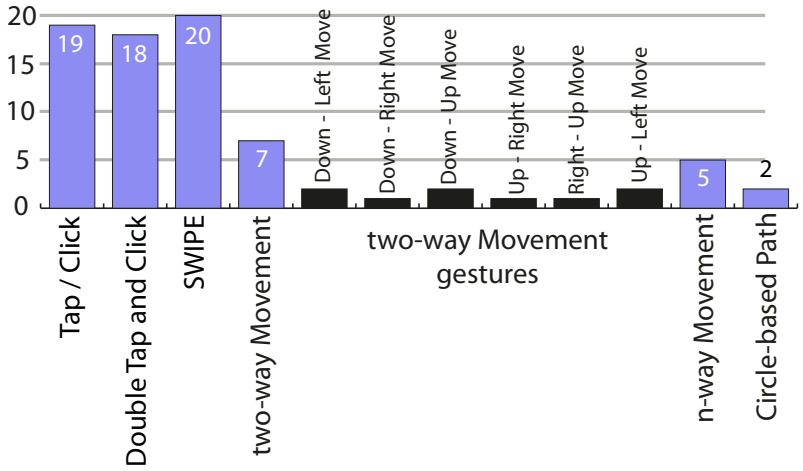


Figure 3.3: Bar chart of all level 1 gestures

software than Autodesk Maya is using the Marking Menu guidance concept for two and n-way movement gestures. Especially n-way gestures are very difficult to memorize, because of their complexity. Interaction Designer should be more willing for simplifying these complex gestures by using a guidance concepts[5]. The section 3.4 about design problems will explain why it is highly recommend to use Marking Menus for n-way gestures.

The last form of gestures in the level 1 category explores circle-based gestures. Only two products use the circle-based gestures. These two products have two completely different purposes for this kind of gesture. BumpTop uses this kind of gestures for executing group selections for items on the screen. The Synaptic trackpad uses their circle-based interaction for scrolling functions. They even created their own term for this gesture, calling it ChiralScrolling™. In general a circle-based gesture is easier to keep in mind for the user in comparison to two and n-way gestures. One disadvantage of circle-based gesture is, that they are not easy to perform, especially with a mouse device. For this reason interaction designers and software engineers should be cautious when using circle-based gestures. The threshold for this gesture must be implemented very well. It could be very disappointing for the user, when they try to perform the gesture without any success.

3.2.2 Level 2 gestures

The level 2 gesture category can be specified in two parts: Direct-Manipulation operations and trained gestures. Direct-Manipulation gestures are the most

[5]A guidance concept is a guide, which appears during the user performs a gesture. That could be predefined gestural guides or dynamic generated gesture suggestions for the user.

supported gestures in the level 2 category.

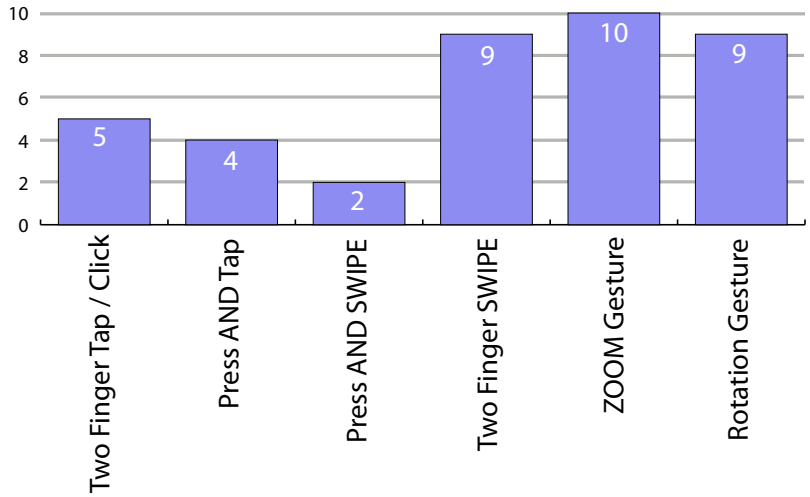


Figure 3.4: Bar chart of all level 2 gestures

They emerged for the mass-market with the advent of the iPhone and other multi-touch systems. The gestures *Zoom* and *Rotation* became famous through the minimal photo editing and browsing video demos on TV or YouTube. For this reason we can already specify these very young gestures (*Two Finger Swipe, Zoom*, and *Rotation* gesture) as intuitive gestures for multi-touch software environments. Interaction designers and software engineers can use without gestures without any doubt. Only in small screen environments, for instance mobile phones, they should consider the small size of the screen and the items. If the graphical representation of a digital item is too small for Direct-Manipulation, they should not use a Direct-Manipulation gesture. Control interfaces like sliders and editing fields are more appropriate in this case. Although rarely used, the *Select (Press) and Swipe* Gesture is even a direct manipulation gesture. The user presses two fingers on the digital object, and then performs with one finger a one-way movement, like a swipe. This gesture can be used for additional cropping and resizing operations (see also at BumpTop).

The next two gestures are trained gestures and not very obvious for the user. The *Two Finger Tap* is supported by five products. Especially, the "right click" metaphor or the command for the context menu is on Windows 7 differently implemented than on Mac OS. On Mac OS the *Two Finger Tap* performs the "right click" metaphor. On Windows 7 systems the user must perform a *Press one Finger and tap with the second finger* gesture for the right-click metaphor. These differences of gesture design for the same command, can confuse the standard computer user. It does not support an optimal independent platform user experience.

3.2.3 Level 3 gestures

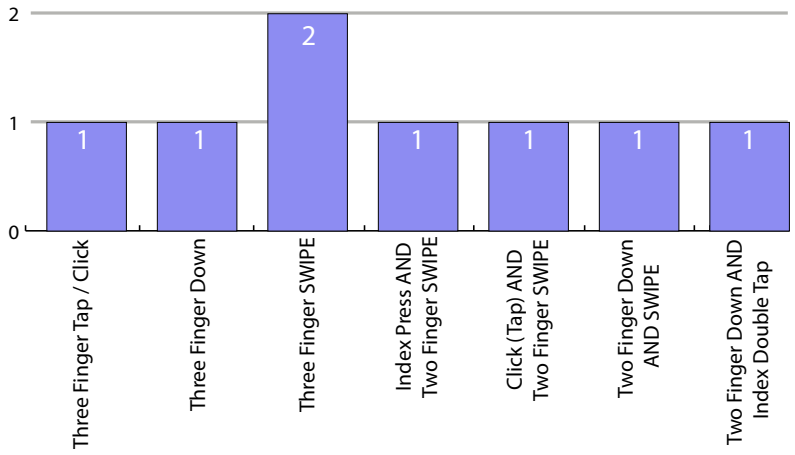


Figure 3.5: Bar chart of all level 3 gestures

Gestures in the level 3 category are not very often used. Only three software products are using these kinds of gestures. The two more or less intuitive gestures *Three Finger Tap* and *Three Finger Swipe* are used in bigger systems (Mac OS and Synaptic Trackpad). The *Swipe* gesture is used again for scrolling and skipping navigation commands in a higher hierarchy. For instance, the Mac Operating System and the Synaptic trackpad use this gesture for skipping between different pages and photos. The other gestures are used only by the Mac Jitouch gesture extension software. The amount of gestures in this collection is quite impressive. Unfortunately the gestures are very difficult to remember, because of the inherited coordination complexity with 3 fingers.

3.2.4 Level 4 and level 5 gestures

Only a few gestures exist in the level 4 and level 5 categories. In the level 4 category Mac OS Leopard uses a *4 Finger Swipe* for switching between different active applications. The Mac Jitouch Gesture extension uses a *4 Finger Follow Tap* for minimizing and maximizing a window on the screen. Both gestures are not obvious for the user. The user has to learn them. Therefore these two gestures belong to the trained gesture category. The *4 Finger Follow tap* is especially difficult to memorize and perform.

BumpTop Version 2 is the only software product, which supports a level 5 gesture. The gesture is called *Scrunch*. The user must use the whole hand for this gesture. He or she spreads the finger and then move the fingers together. All object on the screen, which are underneath the hand will be grouped together. This gesture does not seem obvious at the first view, although it belongs to intuitive gesture category. It is a nice example how gestures can improve group-

command interactions. This gesture is very similar with our activity when we group some items on a desk, for instance. The only exception are mobile screens. These screens are usually smaller than an out stretched hand. For this reason, this level 5 gesture won't work with small screens.

3.2.5 Results from a design view

From a interaction design view the analysis showed that gestural interactions are already established in current end-user products. Direct-Manipulation gestures are the most supported and common gestures. Furthermore, the Mac OS and the Synaptic trackpad use the characteristic of levels for hierarchic-based navigation in screen-based environments:

1. One Finger SWIPE for selection and dragging, deleting, move operations
2. Two Finger SWIPE for scrolling
3. Three Finger SWIPE for skip a slide in a presentation, a photo or a page
4. Four Finger SWIPE for switching between different applications

Interaction Designer and Software Engineers should adapt and consider this concept of hierarchic-based commands. Applying this approach for other gestures, can maybe strongly improve gesture based interaction. Another point is, which kind of commands are still insufficiently studied. The figure 3.2.5 points out which gestures have are related to the basic commands of pointing devices in screen-based environments, and which commands are rarely used.

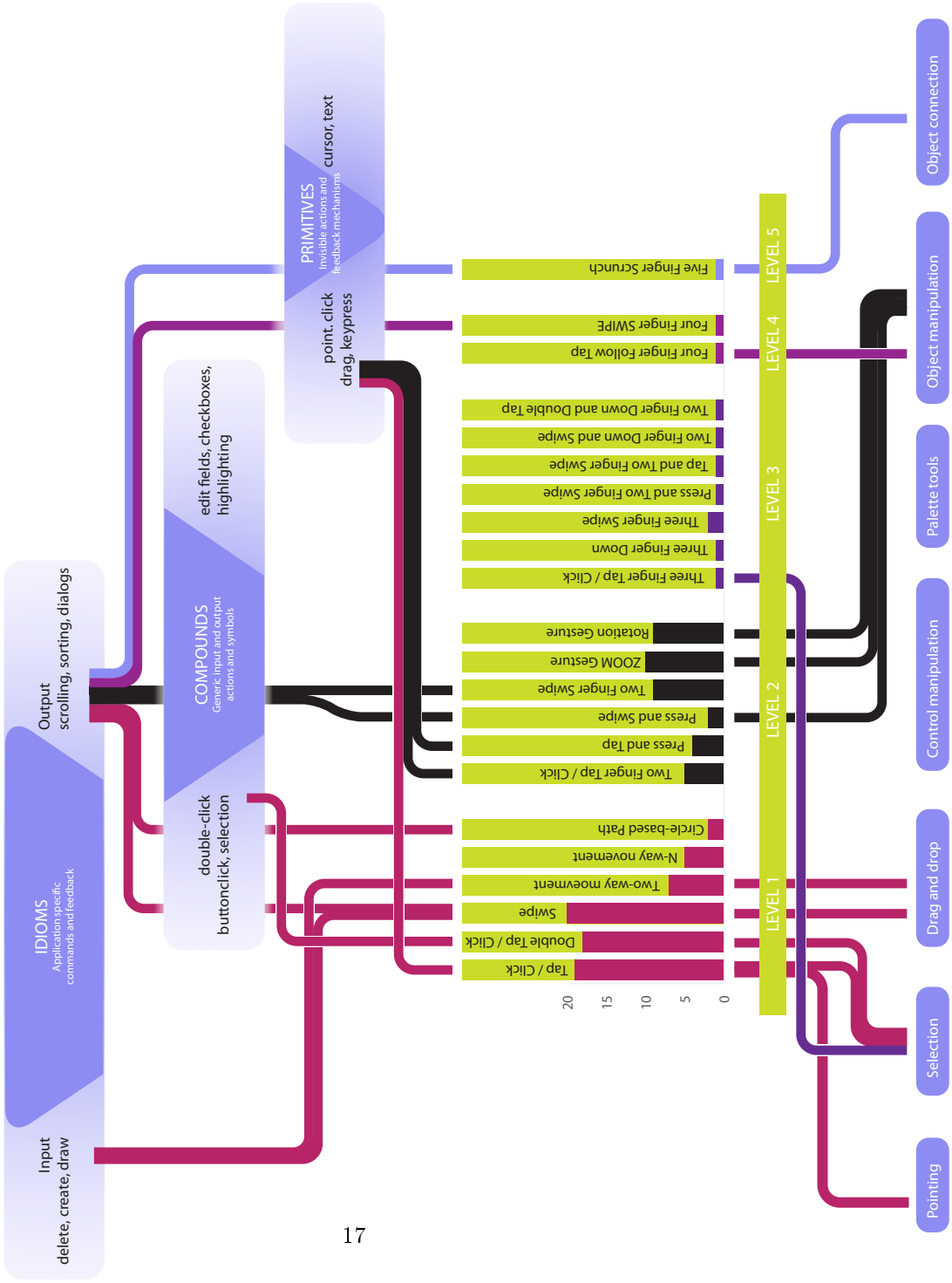


Figure 3.6: Command overview of the investigated gestures

The most basic computer commands for graphical user interfaces are already implemented with gestures. Direct-Object-Manipulation is heavily used because of their intuitive and obvious character for the users. But some operations are still unused in gesture based interaction. For example, the commands related to "Object connections" and "control manipulation" delivers some innovation space for new gesture interactions. The Text Input Software Swype[6] shows object connections in form of character connection gestures in an exemplary way. Palette based commands are heavily connected with the appearance of the cursor. The different cursor styles visualise the active state of the current command for the user. In the domain of Palette tools does not exist much space for innovation of gesture based interaction. Touchscreen environments do not use a mouse pointer any more, and the interaction design is often stateless. In contrast the discipline of tangible interfaces and their new haptic-based tools could improve "palette tool" interactions. In general, the figure also shows that gesture based interaction with pointing devices is well-established. The most basic operations are mapped with gestures and in the future interaction designers and software engineers will develop other new gestures.

3.3 Results from a technical view

3.3.1 Desktop Operating Systems

The Desktop Operating Systems support very similar gesture patterns in their category. The level 1 supported gestures in all Operating Systems are the same. For level 2 gestures it differs a little bit. All Operations Systems support *Rotate* and *Zoom* gestures, even their Programming SDK supports gesture recognition for these gestures. In addition, every Operating System API [7] [8] [9] [10] [11]delivers access to the raw touch data. This raw touch data allows interaction designer and software engineers to develop their own custom gestures.

In general the Operating Systems supports more Direct-Manipulation gestures. These gestures are more intuitive and obvious for the user. For this reason the Operating Systems act more conservative in the topic of gesture-based inter-

[6] Text Input for Screens from Swype Inc. website, http://swypeinc.com/product.html, accessed March 2010

[7] Cocoa Application Framework, from Mac OS X Reference Library website, http://developer.apple.com/mac/library/releasenotes/Cocoa/AppKit.html, accessed February 2010

[8] Windows Touch Gesture Overview, from MSDN Windows Developer Center website, http://msdn.Microsoft.com/en-us/library/dd940543%28VS.85%29.aspx, accessed February 2010

[9] Windows Touch Team, "Touching Windows 7," March 25, 2009, post on blog "Engineering Windows 7," MSDN Blogs, http://blogs.msdn.com/e7/archive/2009/03/25/touching-windows-7.aspx, accessed February 2010

[10] Yochay Kiriaty, "MuliTouch Capabilities in Windows 7", from MSDN Magazine website, http://msdn.microsoft.com/en-us/magazine/ee336016.aspx, accessed February 2010

[11] Apple Inc. "Chapter 2: Life with your MacBook Pro." Manual MacBook Pro. pp. 26-29 (http://manuals.info.apple.com/en_US/MacBook_Pro_13inch_Mid2009.pdf, accessed February 2010)

	Windows 7	Mac OS Leopard	Linux
Click and Tap	x	x	x
Double Click and Tap	x	x	x
SWIPE	x	x	x
Two Finger Tap	x	x	
Press AND Tap	x		
Two Finger SWIPE	x	x	
Zoom	x	x	x
Rotation	x	x	x
Three Finger SWIPE		x	
Four Finger SWIPE		x	

Table 3.1: Gestures of Desktop Operating Systems

action compared to other smaller products, like BumbTop and Autodesk Maya. Unfortunately, at the time of the analysis only a prototype implementation[12] of the new Linux kernel supported native gesture support. In which case please read the information about Linux in this paper critically.

3.3.2 Mobile Operating Systems

The current Mobile Operating systems supports only one pointer, except the Apple iPhone[13] [14] [15]. Therefore the most support gestures in this category are found in the level 1 gesture category. The Mobile Operating Systems API of the Google Android 1.6[16] and Windows Mobile 6.5[17] deliver a gesture programming framework or at least a gesture creation tool. How and what kind of gestures are supported depends heavily on the used mobile hardware and less on the Mobile Operating System. The mobile BlackBerry device Storm 2[18] supports more

[12] Linux native multitouch support. From ENAC Interactive Computing Laboratory website, http://www.lii-enac.fr/en/projects/shareit/linux.html, accessed February 2010

[13] Handling Multi-Touch Events. From iPhone OS Reference Library website, http://developer.apple.com/iphone/library/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/EventHandling/Ev
CH9-SW11, accessed February 2010

[14] Apple Inc. "Chapter 3: Designing an iPhone Application: From Product Definition to Branding." iPhone Human Interface Guidelines. pp. 42 (http://developer.apple.com/iphone/library/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf, accessed February 2010)

[15] Apple Inc. "Chapter 2: Basics." iPhone User's Guide". pp. 18-20 (http://manuals.info.apple.com/en/iphone_user_guide.pdf, accessed February 2010)

[16] Gestures. From Android Developer website, http://developer.android.com/resources/articles/gestures.html, accessed February 2010.

[17] Using Gestures in Windows Mobile 6.5, From MSDN Windows Mobile Developer Center website, http://msdn.microsoft.com/en-us/library/ee220920.aspx, accessed February 2010.

[18] BlackBerry. "BlackBerry Storm 2: Gestures & Shortcuts", (http://erictric.com/wp-content/uploads/2009/10/bb_storm2_gestures.pdf, accessed February 2010)

	Windows Mobile 6.5	Android 1.6	BlackBerry Storm 2	Symbian OS	iPhone
Click / Tap	x	x	x	x	x
Double Click and Tap	x	x	x	x	
SWIPE	x	x	x	x	x
Two-way movement	x	x		x	
n-way movement	x	x			
Zoom					x

Table 3.2: Gestures of Mobile Operating Systems

gestural interaction than the other BlackBerry devices. Symbian OS supports only the basic gestures[19]. In the future the gesture interaction for mobile phones will rapidly increase, because the Android 2.1[20] and Windows Phone 7 are going to support multitouch gestures for their next generation devices.

3.3.3 3rd Party software

The 3rd Party software Flash Player[21], Silverlight[22][23] and JavaFX[24] support gestures. The supported gestures are based on the used Desktop Operating Systems. This relation can be explained by the technology dependency with the Operating Systems. Each of the 3rd party tools uses the supported gestures of the Operating Systems, and pass them through to their own API. That is the reason why these tools support some gestures only for a special platforms. How already mentioned middleware tools like reacTIVision,NUI Group Library, SPARS-UI, and MT4j are not included in the analysis. Even Sparsh UI and MT4j have already implemented their own gesture recognition system. These

[19]Video Player Package/feature playback view gesture support, From Symbian Developer Wiki website, http://developer.symbian.org/wiki/index.php/Video_Player_Package/feature_playback_view_gesture_support, accessed February 2010

[20]Android 2.1 SDK. From Android Developer website, http://developer.android.com/sdk/android-2.1.html, accessed February 2010.

[21]Christian Cantrell; "Multi-touch and gesture support on the Flash Platform", From Adobe Developer Connection website, November 17, 2009, hhttp://www.adobe.com/devnet/flash/articles/multitouch_gestures.html, accessed February 2010

[22]Tim Heuer, "Silverlight 3 Multi-touch: The Basics," July 30, 2009, post on blog "Method of failed," Tim Heuer Blog, http://timheuer.com/blog/archive/2009/07/30/silverlight-3-multi-touch-introduction-fundamentals-basics.aspx, accessed February 2010

[23]Jesse Bishop, "Multi-touch Gesture Recognition in Silverlight 3," November 5, 2009, post on blog "jebishop.blog," http://www.jebishop.com/2009/11/05/multi-touch-gesture-recognition-in-silverlight-3/, accessed February 2010.

[24]Ritter, Simon & Caicedo, Angela . "Build Your Own Multi-Touch Interface with Java and JavaFX Technology." PowerPoint presentation to JavaOneTM Conference, 2008

	Flash Player 10.1	Silverlight 3	Java FX
Click / Tap	x	x	x
Double Click / Tap	x	x	x
SWIPE	x	x	x
Two Finger Tap	x	x	
Press AND Tap	x (only Win)	x (only Win)	
Two Finger Swipe	x (only Win)	x (only Win)	
Zoom	x	x	x
Rotation	x	x	x

Table 3.3: Gestures of 3rd Party software

tools are not used in end-user products. They are more used for implementing prototypes in scientific environments. For this reason they are not involved in the analysis.

3.3.4 Additional Software products

A few of the investigated additional software products use their own gesture logic. If you compare this gesture table with the other gesture tables, then it becomes obvious the additional software category uses the biggest set of gestures. But quantity is not equal to quality in most cases. Most gestures seem to be very experimental. Which is a very good environment to explore new innovative gestures. Some gestures will be useless, but other gestures can evolve to useful gestures. Therefore, the gestures of this category are investigated in a more detail.

The Internet Browser Software Opera[25] and Mozilla Firefox[26][27] are using mouse gestures for navigation commands. These gestures mainly meet the requirements of level 1 two-way gestures. The user has to press the right mouse button and move the mouse along a certain path. After the movement the user releases the right mouse button for performing the command. A certain amount of one and two-way gestures are easy to perform for the user. Three or n-way Movement gestures are very complicated to remember and are usually difficult to perform. Therefore Autodesk Maya 2010[28] uses the Marking Menu guidance concept for their movement gestures. The user gets a very good guidance for performing the appropriate mouse gesture. The size and the movement of the

[25] Mouse Gestures in Opera. From Opera Software website, http://www.opera.com/browser/tutorials/gestures/, accessed February 2010

[26] Mouse Gestures. From Mozilla Developer website, http://optimoz.mozdev.org/gestures/, accessed February 2010

[27] Supported Gestures. From Mozilla Developer website, http://optimoz.mozdev.org/gestures/defaultmappings.html, accessed February 2010

[28] Marking Menu Editor. From Autodesk Maya 2010 manual website, http://download.autodesk.com/us/maya/2010help/index.html?url=WS1a9193826455f5ff-3a29af00119afd28e95-934.htm,topicNumber=d0e104204, accessed February 2010.

gesture could easily changed by the user.

The Mouse gesture Starter[29][8], BumpTop[30] and Mac Jitouch[31] software products extend the gesture interaction of the used Operating Systems. For this reason they do not support the basic gestures like a *Zoom* and *Rotation* gestures. Especially, the Mac Jitouch uses a bunch of new gestures (especially in the level 3 category). The Mac Jitouch gestures are absolutely not intuitive and obvious for the user. All these gestures of the Jitouch software belongs to the category of trained gestures. The user has to invest time for learning them. The software must also deliver a motivation for the user to learn them. Even the Mac Jitouch gesture collection does not meet these requirements, interaction designers and software engineers can gain inspiration from this gesture design.

In the end it is good to see that these special software products experiment with gestural interaction. They are pushing the development of gestural interaction further forward. Not every implementation is good, but it is an important visionary effort for the bigger software products like the Desktop Operating Systems and 3rd Party Tools.

[29] mgLaunch - Mouse Gesture Application Launcher, From Mouse Gesture. website, http://www.mouse-gesture.com/products/mouse_gesture_application_launcher_mglaunch.html, accessed February 2010

[30] BumpTop Multi-Touch Gestures. From BumpTop website, http://bumptop.com/doc/multi-touch-gestures/, accessed February 2010

[31] Trackpad Gestures. From jitouch 2 website, http://www.jitouch.com/index.php?page=gestures, accessed February 2010

	Browser Opera	Browser Firefox	Autodesk Maya	BumpTop Version 2	Mouse Gesture Starter	Mac Jitouch
Click / Tap	x	x	x	x		
Double Click / Tap	x	x	x	x		
SWIPE	x	x	x	x	x	
Two-way movement	x	x	x		x	
n-way movement		x	x		x	
Circle-based path				x	x	
Press AND Tap						x
Press AND SWIPE				x		x
Two Finger SWIPE				x		x
Zoom				x		
Rotation				x		
Three Finger Tap						x
Index Press AND two finger SWIPE						x
Tap AND two finger SWIPE						x
Two finger down AND SWIPE						x
Two Finger Down AND Index Double Tap						x
Four Finger Follow Tap						x
Five Finger Scrunch				x		

Table 3.5: Gestures of additional software products

3.3.5 Hardware products

External input devices always relate to Operating Systems. In the most cases they extend the input methods of the computer, and afford new interaction forms. Certain input devices are assigned for special software products, others are not. For instance, the Apple Magic Mouse[32] and the ordinary mouse devices extend the input hardware of the used computer. These devices send the input data directly to the Operating System and the Operation System decides how to deal with this data. The Synaptic trackpad [33][34] and the Wacom Bamboo Touch [35] & Fun[36] work different. These devices require a special driver software. This driver software decides how to deal with the input data. For instance, the Wacom Bamboo gestures work only in combination with special software products and don't work with software like the Flash Player, Silverlight and JavaFX. Maybe this approach will change in the future. The driver software will use the Operating System gestures API and pass them directly to the Operating System. Afterwards every external multi-touch device should work with every multi-touchable software. From the interaction design view the external input devices support pre-trained gestures, like *click*, *double click*, one-way movement gesture, and the intuitive Direct-Manipulations gestures.

	Synaptic Trackpad	Wacom Bamboo	Apple Magic Mouse	Ordinary Mouse
Click / Tap	x	x	x	x
Double Click / Tap	x	x	x	x
SWIPE	x	x	x	x
Circle-based Path	x			
Two Finger Tap		x		
Two Finger SWIPE	x	x	x	
Zoom	x	x		
Rotation	x	x		
Three Finger Down	x			
Three Finger SWIPE	x			

Table 3.6: Gestures of Hardware devices

[32] Magic Mouse. From Apple Inc. website, http://www.apple.com/magicmouse/, accessed February 2010

[33] Synaptics Gesture Suite™ for TouchPads, From SynapticsTM website, http://www.synaptics.com/solutions/technology/gestures/touchpad, accessed February 2010.

[34] Gestures and Multi-Touch, from SynapticsTM website, http://www.synaptics.com/solutions/technology/gestures, accessed February 2010

[35] Bamboo Touch. From Wacom website, http://www.wacom.com/bamboo/bamboo_touch.php, accessed February 2010

[36] Bamboo Fun. From Wacom website, http://www.wacom.com/bamboo/bamboo_fun.php, accessed February 2010

3.4 General results

Near the end of the analysis we proved that gesture based interaction is already well established in screen based environments. The most supported gestures are Direct-Manipulation gestures. These gestures are more intuitive and obvious for the user in screen-based environments than other commands.

The relation between basic computer commands (see Coopers list of basic commands for pointing devices2.3) and gestures pointed out, that there still exist some innovation possibilities for developing new gestures. Especially, **object connection gestures** which are not well investigated yet. Project Management Tools and Mind-mapping Tools are using "Object Connection" commands heavily. It would be very interesting to assign appropriate gestures for connecting these different objects together and manipulating them. The Text Input software from Swype Inc.[37] is a very nice example for Object Connection gestures. Swype Inc. connects different character together an generates a word out of it. It works similar to the T9 Text Input guidance on mobile phones.

Also the **hierarchic-gestures concept**, which Apple is using for scrolling (*two Finger Swipe*), skipping (*three Finger Swipe*) and switching between active applications (f*our Finger Swipe*), is an interesting approach. The hierarchic swipe gestures command fits very well with navigation interaction. There may exist other human-computer-interaction forms, where it makes sense to use this kind of hierarchical gesture commands.

Interaction Designers and engineers need to find more such intuitive and sustainable gestures approaches. Direct-Manipulation gestures (*Zoom*, *Rotation*, etc.) and hierarchic gestures meet the requirements of the **user-mental-model**[4, page 27-32]. The user is able to merge these gesture concepts with real world behavior. Especially the higher level (multi-point / multi-touch) interaction are not very well investigated yet. A deeper research may pay out on this topic.

The next problem deals with the category of **trained gestures**. In general, Interaction Designer should avoid this kind of gestures for novice user. Trained gestures are comparable with shortcuts on keyboards. The user has to learn them and they are difficult to remember. Therefore a **guidance concept** for these kind of gesture would improve the learning curve dramatically. Fortunately, for pointing devices exist a very well investigated guidance concept called **Marking Menus**. The Marking Menu approach was initially invented for one point devices (stylus and mouse). A Marking Menu extends the functionality of a context menu with the aspect of radial menus. It is a able to perform a command via moving the cursor or the stylus in a certain direction[11, 12, 19, 1]. Marking Menus show every time the user with a radial or flower menu what functions are available to the user. This approach is very useful for 2-way or

[37] Text Input for Screens from Swype Inc. website, http://swypeinc.com/product.html, accessed March 2010

n-way gestures. If the user can not remember the gesture the Marking Menu suggests the next possible movements. So the user can learn the gesture better. After a while he or she can perform the gesture very fast without any guidance. The software product Autodesk Maya uses this approach successfully. Therefore it is highly recommend to use the Marking Menu approach for level 1 two-way and n-way gestures.

The same circumstances are valid for higher level gestures. In general three finger gestures are already quite difficult to remember. The intuitive gestures *Three Finger Swipe*, *Three Finger Tap* and *Three Finger down* are an exception in this category. All other gestures are supposed to provide a guidance concept. Especially the software Mac Jitouch uses many complicated level 3 gestures. These gestures are not poorly designed, but they are more difficult to remember than to perform. The Marking Menu approach can also help in this case. Every gesture which uses more points (fingers) and a *SWIPE* gesture can use this concept. It can suggest which SWIPE direction executes which command. The Marking Menu research in muli-touch environments is not well researched yet. Also providing a guidance concept for more complicated multi-touch gestures is in the early days.

The lack of guidance concepts in gesture based interaction is not the only main problem. During the analysis an additional important issue emerged. Some **gesture documentations** were very difficult to access or poorly documented. The Software Jitouch, BumpTop and the Browser Opera have exemplary documentation of their supported gestures. Unfortunately, at the moment there exist no real conventions how to document gestures and how to use them. Ryan Lee[38], Ron George[39], and Ideum[40] published free graphics for consistent gesture documentation. Unfortunately, the graphics of Ryan Lee and Ideum (GestureWorks) have only a fixed collection of gestures. For this reason this paper uses the approach from Ron George. His documentation graphics support creating new custom gestures. Therefore it seems the best way for documenting gestures in static graphics and print media. Dan Saffer[17] and R. Clayton Miller[41] take it an important step further. They agree with a consist gesture documentation, but they also started a dialogue for a consist gesture usage. The goal of their dialogues is to find gesture patterns. Gesture patterns are like user interface and design patterns, which function the same across different software products. The aim of this goal is that the user will benefit from a consist user experience between different software products and platforms. This paper supports these different dialogues, and hopes to simplify the daily usage of gestures in the future.

From the technical perspective smaller software products are more motivated

[38] GestureCons. From Ryan Lee website, http://gesturecons.com/, accessed March 2010.

[39] GesturCons. From Ron George Blog website, http://blog.rongeorge.com/design/gesturcons/, accessed March 2010

[40] Open Source Multitouch Gesture Library and Illustrations. From GestureWorks website, http://gestureworks.com/about/open-source-gesture-library/, accessed March 2010

[41] 10/GUI. From R. Clayton Miller website, http://10gui.com, accessed March 2010

to use new gesture concepts than the larger software products. Furthermore, some software products need to deal with technical issues. For instance, Mobile Operating Systems could support more gestures but the mobile hardware defines what is possible. The same is valid for 3rd Tool software. The Operating Systems define which gestures are supported by the 3rd Party Tools and which are not. Almost no 3rd Party Tool manufacturer implemented their own gesture recognition system.

In addition, **touch screens** also have disadvantages compared to traditional pointing devices[17, page 26-27]. The **right-click** menu is not really available on touch screens environments. Also the mouse-hover effect disappeared in touch screens environments. For this reason, the interaction designer should consider that most touch screen environments and also gesture-based interactions are **stateless** or **modeless**. That is not a big problem for the user, but the interaction designer should take care of these differences compared to traditional pointing devices (such as the mouse device). In some cases the visual feedback works different than for traditional pointing devices.

Also the implementation of **gesture recognition threshold** depends on the screen type. The pointer of the finger is much bigger than the pointer of a mouse device. Gesture recognition systems must be able to deal with this inaccuracy in touch screen environments. Hrvoje Benko, Andrew D. Wilson and Patrick Baudisch specified some guidance in their paper "*Precise Selection Techniques for Multi-Touch Screens*"[6]. Using a fingertip instead of a mouse pointer or a stylus pen will be more inaccurate.

The next important point is the characteristic of movement in the threshold implementation. Dynamic gestures are more difficult to detect, than the static gestures. Therefore the threshold implementation is crucial factor in gesture recognition systems. In general traditional pointing devices contain a certain inaccuracy in performing a movement. Consequently interaction designers and software engineers need to implement a threshold to their pattern-match gesture system. It is almost impossible for the user to perform a gesture with 100% accuracy. Especially, circle-path based gestures are more difficult to perform with a mouse device than in a touch screen environment. The software BumpTop and the Synaptics trackpad implemented their circle-path gesture pretty well.

The last technical point, **size and resolution of the screen** influence the design process of gestures strongly. Some gestures, for instance, the *Five Finger Scrunch* gesture from BumpTop, are not able to adapt for mobile devices. The screen is too small for using whole hand and the fingers. One possibility might be to reducing the amount of fingers for performing a *Scrunch* gesture. Instead of using 5 fingers use 3 fingers. The disadvantage of this solution is the broken user experience of the gesture. Another point is the different usage context of small screen compared to bigger screens. Small screen devices are also very often used with one hand. An example scenario could be in a bus. The user hold himself with one hand, and the device in the other hand. In these kinds of scenarios

there is only the thumb left for gesture interaction. This circumstance needs a completely different gesture design approach. Also in big screens environments the *Five Finger Scrunch* gesture will not work. For instance, the ring°wall from Sensory Minds[42] uses big graphical items in their user interface. Their graphical items are too big to group them together with one hand. Pointing and Selection via arm gestures could be a solution for this problem[18]. In the end the screen size is not a problem for gesture based interaction. The interaction designer must consider the screen capabilities, and then choose the appropriate gesture design approach.

This chapter explored and highlighted the common gestures. It mentioned the problems with gesture interaction in general, and proposed some solutions. The analysis also showed, that the most used gestures (*Double Click*, *SWIPE*, *Zoom* and *Rotation* gestures) should used in the same manner, as other end-user products are doing. In the end it will provide a better user experience.

[42]SENSORY MINDS - NATURAL USER INTERFACE DESIGN. From Sensory Minds website, http://10gui.com, accessed March 2010

Chapter 4

An Industrial Design view on the evolution of Pointing Devices

After researching the history of pointing devices it seems, that interaction designers are constantly changing their hardware approach to a software approach. This evolution is interesting from an industrial designer view. Industrial designers will have less influence when designing a device, or industrial designers must also learn the principles of Interface Design.

Figure 4 points out this basic idea of evolution for Desktop PC and Notebooks.

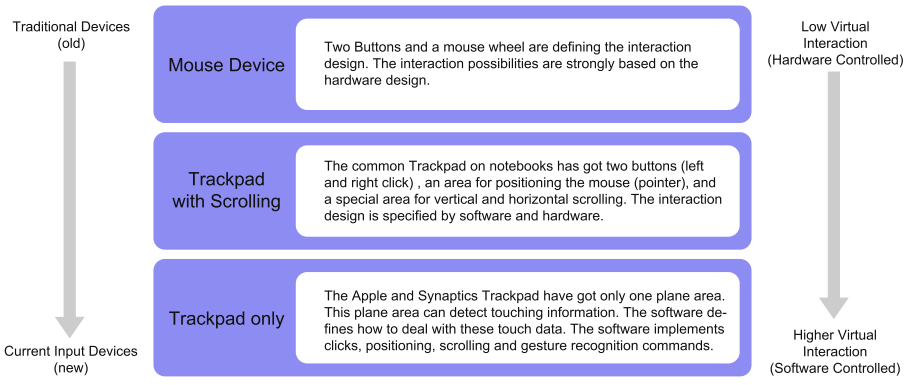


Figure 4.1: Evolution of Laptop Pointing Devices

A very similar evolution emerged for cellphone devices. The first cellphone devices are strongly based on keyboard interaction. Then the cellphone manufacturer added a joystick to their devices for a better vertical and horizontal

navigation. Afterward the pen (stylus) interaction became more popular. Until now the keyboard is almost completely gone (see figure 4). The iPhone and the Google Phone is heavily using the touchscreen for direct interaction with the device.

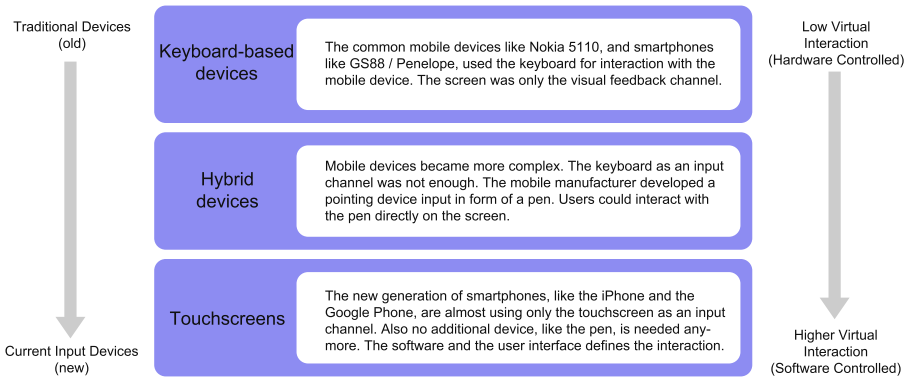


Figure 4.2: Evolution of mobile devices

Are these two developments evidence for the hypothesis, that the input channel for pointing devices become more virtual? Will be the physical inputs as buttons, joysticks, and (mouse) wheels be completely removed in the future? Will be the physical parts be replaced by virtual inputs via touchscreens? The direct interaction with the screen emphasizes the idea of Direct-Manipulation. Direct-Manipulation is easier to learn and more intuitive for humans. It addresses how a human works directly with physical things in the real world. The physical inputs can not really realize this Direct-Manipulation approach in screen-based environments.

Unfortunately in this paper it was not possible to find an evidence, that Direct-Manipulation via touch or pointing device performs better, than a physical input system.

In relation to virtual inputs and their evolution, it is obvious that gesture based interaction is going to play a very important role. The success of the iPhone and other multi-touch systems confirm this development in general. The future of gesture interaction doesn't rely only on touch gestures. Also free-form gestures are playing an important role. Especially in the game industry. The success of the Nintendo Wii[1] is an evident for this evolution. Also the camera based interaction system Project Natal[2] from Microsoft and the BiDi Screen[3] from the MIT confirm the evolution of gesture based interaction.

[1] Nintendo Wii. From Wii.com. website, http://www.wii.com/, accessed March 2010

[2] Project Natal. From Microsoft Xbox.com. website, http://www.xbox.com/en-US/live/projectnatal/, accessed March 2010

[3] BiDiScreen, From MIT Media Lab. website, http://web.media.mit.edu/~mhirsch/bidi/, accessed March 2010

For this reason it is appropriate to ask if the hardware based input will completely disappear. Will humans interact with computer without any device in the future?

This free form of interaction has one disadvantage. The user needs a feedback if the gesture or interaction is successfully performed. If not, then the user needs to be informed about the wrong performance. Tangible (physical) devices can deliver such guidance. The field of organic and kinetic interfaces[14, 5, 7, 13, 16, 20] can solve this task. They deal with the tangible characteristic of interaction and feedback. These devices decrease the user's freedom of movement. It is important to understand that this research is not against free-form (gesture) interaction. In some cases kinetic based interaction and free form interaction can benefit from each other. Kinetic devices can give the user a guidance or at least a visual feedback for the correct gesture performance. It supports the users movement. Moreover the project D20 from Olivier Bau[4] and Poupyrev[15] shows the combination of multi-touch gestures and a tangible device in an exemplary form. Therefore these two different Human Computer Interaction approaches will merge in the future. The Wiimote device and its vibration feedback channel combined with the gesture based interaction is only the beginning of this evolution. The future lies in the combination of Industrial and Interface Design. Industrial designers have the knowledge about the human kinetic and movement. interface designers have the knowledge about the visual and information perception. Both disciplines are vital for developing next generation devices.

[4] D20: Interaction with Multifaceted Display Devices, From Olivier Bau Institute website, http://insitu.lri.fr/~bau/d20.html, accessed March 2010

Chapter 5

Conclusion

This paper showed that gesture based interaction is a very powerful tool in Human Computer Interaction. It has already arrived in large software products, such as Desktop Operating Systems. It is also obvious that gesture based interaction with pointing devices can improve and speed up the interaction between humans and computers[9]. Gestures do not create a new communication form or even a new interaction language. Until the merge between organic interfaces and free form gestures is complete, gesture-based interaction with pointing devices will aid in the transition between these different Human Computer Interaction approaches.

A resumption of this paper could be an investigation in gestures depending on the screen size. In the future it is important to find the answer for the questions:

- Which gestures are appropriate for which screen size?
- How do the gesture-based interactions distinguish each other from different screen-based environments?

The next point could be in researching the media content of games. Games support highly interactive content and user interfaces. It is also interesting to know which gestures are used in games. Which gesture are useful and how they distinguish from each other in different game genres. Strategy games might use other gestures than Ego-Shooter games. Which genre-based gestures exist for games? It would be also very interesting to explore the free-form gesture and pointing device gesture patterns in games. Are free-form gestures more used than pointing device gestures in games? Maybe the results will inspire the future gestures in common software products (like in Mind-mapping Tools).

There still exist many different domains to investigate gestural interaction. This paper summarized only a small part of this domain. The results gave an overview what is actually state of the art in gesture interaction for pointing

devices. Now we should accurately observe the new developments in free-form and multi-touch gestures.

Bibliography

[1] Lecolinet E Bailly, G. and L. Nigay. Flower menus: A new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. In *AVT2008*, pages 15–22, Napoli, Italy, May 2008.

[2] Mark Billinghurst and Bill Buxton. Human input to computer systems: Theories, techniques and technology. Chapter 14: Gesture Based Interaction, 2002.

[3] Bill Buxton. Multi-touch systems that i have known and loved. Bill Buxton Research Website, January 2010.

[4] Cronin Cooper, Reimann. *In About Face 3 - The Essential of Interaction Design*. Wiley Publishing, 2007.

[5] David Holman and Roel Vertegaal. Organic user interfaces: designing computers in any way, shape, or form. *Commun. ACM*, 51(6):48–55, 2008.

[6] Andrew D. Wilson Hrvoje Benko and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *CHI 2006 Proceedings - Interacting with Large Surfaces*, pages 1263–1272, Montréal, Québec, April 2006. CHI 2006.

[7] Hiroshi Ishii. The tangible user interface and its evolution. *Commun. ACM*, 51(6):32–36, 2008.

[8] Yang Li Jacob O. Wobbrock, Andrew D. Wilson. Gestures without libraries, toolkits or training: A one dollar recognizer for user interface prototypes. In *UIST 2007*, Newport, Rhode Island, USA, 2007. UIST.

[9] Kenrick Kin, Maneesh Agrawala, and Tony DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *GI '09: Proceedings of Graphics Interface 2009*, pages 119–124, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[10] E. Kurtenbach, G. & Hulteen. *The Art of Human Computer Interface Design*, chapter Gestures in Human-Computer Communications, pages 309–317. Addison-Wesley, 1990.

[11] W. Kurtenbach, G. & Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of InterCHI*, pages 482–487, 1993.

[12] W. Kurtenbach, G. & Buxton. User learning and performance with marking menus. In *Proceedings of CHI*, pages 258–264, 1994.

[13] Kas Oosterhuis and Nimish Biloria. Interactions with proactive architectural spaces: the muscle projects. *Commun. ACM*, 51(6):70–78, 2008.

[14] Amanda Parkes, Ivan Poupyrev, and Hiroshi Ishii. Designing kinetic interactions for organic user interfaces. *Commun. ACM*, 51(6):58–65, 2008.

[15] Ivan Poupyrev, Henry Newton-Dunn, and Olivier Bau. D20: interaction with multifaceted display devices. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 1241–1246, New York, NY, USA, 2006. ACM.

[16] Jun Rekimoto. Organic interaction technologies: from stone to skin. *Commun. ACM*, 51(6):38–44, 2008.

[17] Dan Saffer. *Designing Gestural Interfaces.* O'Reilly, 2009.

[18] Alexander Schick, Florian van de Camp, Joris Ijsselmuiden, and Rainer Stiefelhagen. Extending touch: towards interaction with large-scale surfaces. In *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 117–124, New York, NY, USA, 2009. ACM.

[19] Ravin Balakrishnan Shengdong Zhao. Simple vs. compound mark hierarchical marking menus. In *UIST*, pages 33–42, Santa Fe, New Mexico, USA,, October 2004.

[20] Roel Vertegaal and Ivan Poupyrev. Introduction. *Commun. ACM*, 51(6):26–30, 2008.

Appendix

In this appendix all gestures are listed with a description, the supported command, and the supported product. The gesture graphics are based on the gesture documentation of GesturCons from Ron George[1].

[1] GesturCons. From Ron George Blog website, http://blog.rongeorge.com/design/gesturcons/, accessed March 2010

Tap or Click

The user presses a mouse button and releases the same mouse button fastly. For touch device the user touches the screen at a certain position, and removed the finger directly after the touch contact.

Level 1

Gesture graphic	Gesture Name	Command	Products
	Tap and Click	Select	Windows 7
	Tap and Click	Select	Mac OS
	Tap and Click	Select	Linux
	Tap	Select	Windows Mobile 6.5
	Tap	Select	Google Android 1.6
	Tap	Select	BlackBerry Storm 2
	Tap	Select	Symbian OS
	Tap	Select	iPhone
	Tap and Click	Select	Adobe Flash
	Tap and Click	Select	Silverlight
	Tap and Click	Select	JavaFX
	Tap	Select	Wacom Bamboo
	Tap	Select	Synaptic Trackpad
	Tap	Select	Mac Trackpad
	Click	Select	Mice Device
	Tap and Click	Select	Apple Magic Mouse
	Tap and Click	Select	Opera Browser
	Tap and Click	Select	Firefox Browser
	Tap and Click	Select	Autodesk Maya
	Tap and Click	Select	BumpTop Version 2
	Click	Select	Mouse Gesture Start
	Tap	Select	Mac Jitouch

Level 2

Gesture graphic	Gesture Name	Command	Products
	Two Finger Tap	custom	Windows 7
	Two Finger Tap	Right Click Menu (Context Menu)	Mac OS
	Two Finger Tap	custom	Adobe Flash (only on Windows)
	Two Finger Tap	custom	Silverlight
	Two Finger Tap	Right Click Menu (Context Menu)	Wacom Bamboo
	Two Finger Tap	Right Click Menu (Context Menu)	Mac Trackpad

Level 3

Gesture graphic	Gesture Name	Command	Products
	Three Finger Tap	Open new link in a tab	Mac Jitouch

Double Tap or Double Click

The user fastly click the mouse button two times. For touch device the user touched the screen at a certain position, and removed the finger directly after the touch contact. This Tap action the user performs two times.

Level 1

Gesture graphic	Gesture Name	Command	Products
	Double Tap and Double Click	Open, Activate	Windows 7
	Double Tap and Double Click	Open, Activate	Mac OS
	Double Tap and Double Click	Open, Activate	Linux
	Double Tap	Open, Activate	Windows Mobile 6.5
	Double Tap	Open, Activate	Google Android 1.6
	Double Tap	Open, Activate	BlackBerry Storm 2
	Double Tap	Zoom	Symbian OS
	Double Tap and Double Click	custom	Adobe Flash
	Double Tap and Double Click	custom	Silverlight
	Double Tap and Double Click	Open, Activate	JavaFX
	Double Tap	Open, Activate	Wacom Bamboo
	Double Tap	Open, Activate	Synaptic Trackpad
	Double Tap	Open, Activate	Mac Trackpad
	Double Click	Open, Activate	Mice Device
	Double Tap and Double Click	Open, Activate	Apple Magic Mouse
	Double Tap and Double Click	Open, Activate	Opera Browser
	Double Tap and Double Click	Open, Activate	Firefox Browser
	Double Tap and Double Click	Open, Activate	Autodesk Maya
	Double Tap and Double Click	Open, Activate	BumpTop Version 2
	Double Click	Open, Activate	Mouse Gesture Start
	Double Tap	Open, Activate	Mac Jitouch

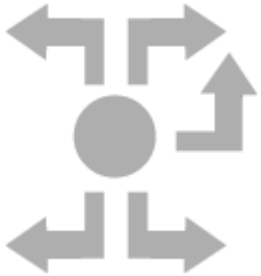
Swipe

The user presses a mouse button, then moving the mouse cursor to a certain position, or fast anywhere on the screen. After the moving action the user releases the mouse button. For touch device the user touches the screen at a certain position (graphic object or a special area), and moves the finger along the screen. The end of the movement can be a certain position on the screen, or somewhere on the screen.

Level 1

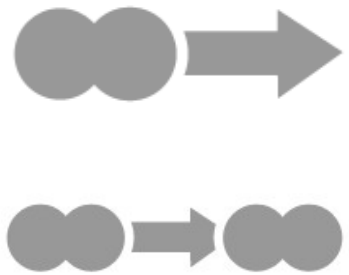
Gesture graphic	Gesture Name	Command	Products
	Swipe, Flick	Drag, Drag and Drop, Scrolling	Windows 7
	Swipe	Drag, Drag and Drop	Mac OS
	Swipe	Drag, Drag and Drop	Linux
	Swipe	Drag, Drag and Drop, Scrolling	Windows Mobile 6.5
	Swipe	Drag, Drag and Drop, Scrolling	Google Android 1.6
	Swipe	Drag, Drag and Drop, Scrolling	BlackBerry Storm 2
	Swipe	Drag	Symbian OS
	Swipe	Drag, Scrolling, Skipping	iPhone
	Pan	custom	Adobe Flash
	Swipe	custom	Silverlight
	Swipe	Drag	JavaFX
	Swipe	Drag, Scrolling	Wacom Bamboo
	Momentum	Drag, Drag and Drop	Synaptic Trackpad
	Swipe	Drag, Drag and Drop	Mac Trackpad
	Swipe	Drag, Drag and Drop	Mice Device
	Swipe	Drag, Drag and Drop, Scrolling	Apple Magic Mouse
	Swipe	• Left mouse for dragging and drag • Right Click for gesture performing	Opera Browser
	Swipe	• Left mouse for dragging and drag • Right Click for gesture performing	Firefox Browser

	Swipe	• Left mouse for dragging and drag • Right Click for gesture performing	Autodesk Maya
	Flick	Drag, Drag and Drop	BumpTop Version 2
	Swipe	• Left mouse for dragging and drag • Right Click for gesture performing	Mouse Gesture Start

Gesture graphic	Gesture Name	Command	Products
	Two way gesture	custom	Windows Mobile 6.5
	Two way gesture	custom	Google Android 1.6
	Two way gesture	custom	Symbian OS
	Two way gesture	Maximize window, close tab, open tab, etc.	Opera Browser
	Two way gesture	Maximize window, close tab, open tab, etc.	Firefox Browser
	Two way gesture	custom	Autodesk Maya
	Two way gesture	custom	Mouse Gesture Start

Gesture graphic	Gesture Name	Command	Products
	n-way gesture	custom	Windows Mobile 6.5
	n-way gesture	custom	Google Android 1.6
	n-way gesture	custom	Symbian OS
	n-way gesture	custom	Firefox Browser
	n-way gesture	custom	Autodesk Maya
	n-way gesture	custom	Mouse Gesture Start

Level 2

Gesture graphic	Gesture Name	Command	Products
	Two finger swipe	Scrolling	Windows 7
	Two finger swipe	Scrolling	Mac OS
	Two finger Pan	custom	Adobe Flash
	Two finger Pan	custom	Silverlight
	Two finger swipe	Scrolling, skipping	Wacom Bamboo
	Two finger swipe	Scrolling	Synaptic Trackpad
	Two finger swipe	Scrolling	Mac Trackpad
	Two finger swipe	Skipping	Apple Magic Mouse
	Fan Out, Pan, Focus on Wall	Ungrop, drag, select	BumpTop Version 2

Level 3

Gesture graphic	Gesture Name	Command	Products
	Three finger swipe	Skipping	Mac OS
	Three finger swipe	Skipping	Synaptic Trackpad
	Three finger swipe	Skipping	Mac Trackpad

Level 4

Gesture graphic	Gesture Name	Command	Products
	Three finger swipe	Switching beween Applications	Mac OS
	Three finger swipe	Switching beween Applications	Mac Trackpad

Press and Tap

For touch device the user touches the screen with one finger (e.g. Index finger) at a certain position, and with the second finger (e.g. Middle finger) the user performs a short tap

Level 2

Gesture graphic	Gesture Name	Command	Products
	Press and Tap	Right click menu (context menu)	Windows 7
	Press and Tap	custom	Adobe Flash (only on Windows)
	Press and Tap	custom	Silverlight

Zoom

For touch device the user touches the screen with two fingers. Afterwards move the two fingers together or away from each other.

Level 2

Gesture graphic	Gesture Name	Command	Products
 	Zoom	Scaling	Windows 7
	Pinch	Scaling	Mac OS
	Zoom	Scaling	Linux
	Pinch	Scaling	iPhone
	Zoom	Scaling, custom	Adobe Flash
	Zoom	Scaling, custom	Silverlight
	Zoom	Scaling	JavaFX
	Zoom	Scaling	Wacom Bamboo
	Zoom	Scaling	Synaptic Trackpad
	Pinch	Scaling	Mac Trackpad
	Zoom, Grow and shrink	Scaling	BumpTop Version 2

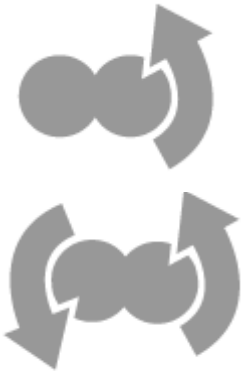
Rotate

For the level 1 gesture. Press with one finger the mouse button or on the screen. Then perform with the mouse cursor or your finger a circle path movement. For touch devices the user touches the screen with two fingers (2 level). Afterwards rotate the two fingers together clockwise or counter-clockwise.

Level 1

Gesture graphic	Gesture Name	Command	Products
	ChiralScrool™	scrolling	Synaptic Trackpad
	Lasso, Lasso'n'Cross	Grouping and ungrouping	BumpTop Version

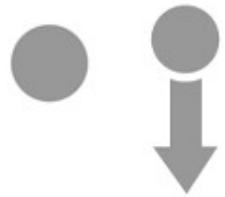
Level 2

Gesture graphic	Gesture Name	Command	Products
	Rotate	No definition	Windows 7
	Rotate	rotate	Mac OS
	Rotate	rotate	Linux
	Rotate	custom	Adobe Flash
	Rotate	custom	Silverlight
	Rotate	rotate	JavaFX
	Rotate	rotate	Wacom Bamboo
	Rotate	rotate	Synaptic Trackpad
	Rotate	rotate	Mac Trackpad
	Rotate	Rotate 3D View	BumpTop Version 2

Finger(s) down and Swipe

The user press one or more fingers down and then performs a swipe gesture with one or more fingers.

Level 2

Gesture graphic	Gesture Name	Command	Products
	Finger (index finger) down and Swipe	Cropping a photo	BumpTop Version 2
	Finger (index finger) down and Swipe	drag and scaling	Mac Jitouch

Level 3

Gesture graphic	Gesture Name	Command	Products
	Finger (index finger) down and two-finger swipe	Open and close tab	Mac Jitouch

Gesture graphic	Gesture Name	Command	Products
	Two fingers down and one finger swipe	Skipping (Switching to the chosen neighbor space)	Mac Jitouch

Three fingers down

The user presses with three fingers on the touch sensitive area.

Level 3

Gesture graphic	Gesture Name	Command	Products
	Three fingers down	Opening applications	Mac Jitouch

Tap and Swipe

The user performs a tap gesture and directly after that a swipe gesture is performed.

Level 3

Gesture graphic	Gesture Name	Command	Products
	Tap and two-finger swipe	Quit an application	Mac Jitouch

Double tap and finger down

The user presses two fingers (e.g. Middle and ring finger) down and then performs with one finger (e.g. Index finger) a double tap.

Level 3

Gesture graphic	Gesture Name	Command	Products
	Double Tap and two-finger down	Refresh, update	Mac Jitouch

Following fingers down

The user starts pressing with one finger (e.g. Index finger) on the touch-sensitive area and then does the same with the next three fingers.

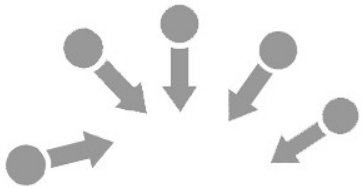
Level 4

Gesture graphic	Gesture Name	Command	Products
	Four finger follow press	Maximize and Minimize the window	Mac Jitouch

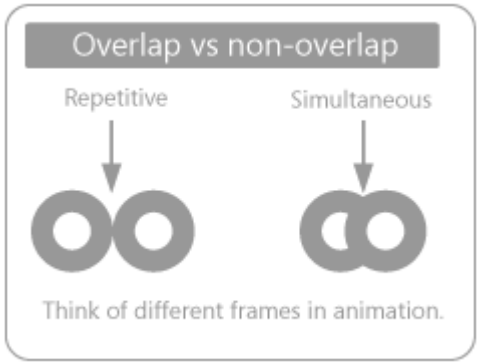
Scrunch

The user touch with all fingers of one hand on the touch sensitve area, and moves the finger together

Level 5

Gesture graphic	Gesture Name	Command	Products
	Five finger scrunch	grouping	BumpTop Version 2

Gesture Language based on Gesturcons



Rotate :: User places contact and rotates contact.

Twin Hold Rotate:: User places two contacts on device, one contact rotates.

Twin Hold Twin Rotate :: User places two contacts on device, both contacts rotate.

Path :: User places contact on device and follows a predetermined path that must be adhered to.